UNIVERSIDADE NOVA DE LISBOA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA

Visitas guiadas através de modelos de volumetria

Sofia Ester Pereira Reis

Dissertação apresentada na Faculdade de Ciências e Tecnologia
da Universidade Nova de Lisboa para obtenção do grau de
Mestre em Engenharia Informática

Orientador: Professor Doutor Manuel Próspero dos Santos

MONTE DE CAPARICA

2006

Sumário

Esta dissertação tem por fim o estudo de algoritmos que permitem efectuar uma visita guiada a um modelo volumétrico.

A visita guiada permite ao utilizador observar todas as partes relevantes do modelo. Assim, expõe-se como seleccionar de forma automática as posições que melhor permitem observar as partes relevantes. Dadas essas posições, mostra-se como criar um percurso que passe por todas elas.

O esforço exigido ao utilizador, durante a visita guiada, face à metodologia aplicada, é o mínimo possível, como se este estivesse numa plataforma móvel e alguém empurrasse essa plataforma. Liberto de todas as preocupações associadas à sua deslocação, o utilizador poderá concentrar-se apenas na observação do modelo.

É também abordado o caso particular da visita guiada a um modelo representativo de um museu de pintura. As técnicas estudadas são implementadas, tendo sido criado um protótipo que permite efectuar uma visita guiada ao museu. Nessa visita guiada, as peças expostas, que são as partes relevantes do museu, são visualizadas da melhor forma possível, ou seja, de maneira a que sua dimensão e o contexto em que estão inseridas sejam perceptíveis para o utilizador.

Abstract

This thesis studies algorithms that allow the user to make a guided tour inside a volumetric model.

During the guided tour, the user observes all the relevant parts of the model. The positions that allow the user to observe the relevant parts are automatically selected. It will be demonstrated, how to select these positions. After the positions have been selected, a tour that passes through all of them will be created.

The effort required from the user, during the guided tour, is the minimal possible, as if the user was placed in a mobile platform and someone pushed that platform. Freed from all the concerns relative to his or her dislocation, the user will be able to focus only in the observation of the model.

The specific case of a guided tour to a pictures museum is also analysed. The algorithms studied are implemented, and a prototype that allows a guided tour to the museum has been created. In this guided tour, the pictures in display are observed in the best possible way, in such a manner that their dimension and context are perceptible to the user.

Agradecimentos

Ao Professor Doutor Próspero dos Santos, pela orientação, pela disponibilidade e pelo apoio.

Aos meus pais, por terem sempre acreditado em mim, mesmo quando eu própria não acreditei.

A todas aqueles, e foram muitos, que me dedicaram o seu apoio generoso e incondicional, ficando detentores da minha amizade e gratidão.

Índice de matérias

Índice de figuras

Índice de quadros

1. Introdução

Nesta tese são estudados os algoritmos que permitem efectuar uma visita guiada a um modelo geométrico que seja um espaço fechado. Esse espaço fechado é um edifício tal como um museu, uma fábrica ou um prédio de escritórios. Caso, devido ao elevado número de divisões e corredores, o modelo seja de carácter complexo, podem surgir diversos problemas entre os quais se destacam os seguintes [AFV04, Vázq03a]:

- O problema mais imediato é a possibilidade de o utilizador se perder no modelo, limitando-se a vaguear de forma aleatória através dele. O utilizador terá maior probabilidade de se perder caso o aspecto do modelo seja uniforme e sem elementos que possam servir como pontos de referência no caminho a percorrer.
  - Exemplo:
    - Se o modelo for um mero labirinto de corredores, todos iguais uns aos outros, um utilizador com um sentido de orientação menos apurado poderá facilmente perder-se.

- Além do risco do utilizador se perder, existe também outro risco, porventura maior. Caso a visita do utilizador não seja orientada, pode acontecer que este deixe passar despercebidas partes interessantes do modelo.
  - Exemplo:
    - Vamos supor que estávamos perante um modelo do Museu Nacional de Arte Antiga. Se a visita a este hipotético modelo não for orientada, o utilizador pode deixar passar despercebidos os Painéis de São Vicente de Fora ou as Tentações de Santo Antão. Ora, as duas obras de arte mencionadas são, certamente, pontos de grande interesse no museu.

- Ainda que o utilizador não se perca, e consiga encontrar as partes relevantes do modelo, pode acontecer que ele não as visualize da melhor forma, ou que não dê a devida atenção a pormenores importantes.
  - Exemplos:
    - Ao visualizar uma pintura, o utilizador deve estar virado de frente para pintura, pois essa é a posição em que lhe é transmitida maior quantidade de informação por parte da obra. Além disso, a distância do utilizador à pintura deve ser óptima. Se o utilizador estiver muito longe, não conseguirá visualizar os detalhes. Se estiver muito perto, não consegue visualizar a pintura toda e perde a noção da globalidade.
    - No caso de uma escultura, o utilizador poderá não a visualizar de todos os ângulos que sejam relevantes, deixando escapar pormenores importantes da obra de arte.

- Os elementos arquitectónicos do modelo e o mobiliário podem dificultar a movimentação no modelo. Um utilizador menos familiarizado com os computadores e, por consequência, menos habituado a deslocar-se através de ambientes virtuais poderá ter dificuldade em encontrar o caminho que o leve até às partes interessantes do modelo.
    - Exemplo:
        - A deslocação através de um modelo repleto de escadas, colunas e corredores estreitos e sinuosos.

Esta dissertação, tal como já foi referido, é dedicada ao estudo dos algoritmos que permitem ao utilizador fazer uma visita guiada a um modelo geométrico que seja um espaço fechado. A geração de visitas guiadas em espaços abertos, tais como uma cidade, fica fora do âmbito desta dissertação.

A visita guiada conduz o utilizador através do modelo permitindo-lhe observar todas as partes deste que sejam interessantes. Além disso, cada parte interessante do modelo deve ser observada através dos melhores pontos de vista possíveis.

O esforço e a interacção exigida ao utilizador deve ser mínimo, para que este se possa concentrar, apenas, na observação das zonas interessantes do modelo. É como se o utilizador estivesse colocado numa plataforma móvel, e alguém empurrasse a plataforma, libertando-o de todas as preocupações com a deslocação.

Uma vez que a visita guiada é efectuada num modelo complexo, essa visita é sujeita aos problemas que foram enunciados anteriormente. Os algoritmos estudados devem conseguir resolver, ou pelo menos atenuar, esses problemas.

Foi também desenvolvido um protótipo que, através da implementação dos algoritmos estudados, consegue guiar o utilizador através do modelo de um museu de pintura.

1.1. Trabalho prévio

No que se refere ao trabalho prévio, realizado dentro desta área, é de destacar a planificação do caminho de um robô. Em [Isto97], Pekka Isto apresenta um algoritmo que opera em dois níveis, recorrendo a uma pesquisa local e a uma pesquisa global.

Tsai-Yen Li e Hung-Kai Ting criaram uma abordagem que melhora a navegação tridimensional, poupando ao utilizador manobras desnecessárias, devidas às colisões. Recorrendo aos dados fornecidos pelo utilizador, através do rato, Tsai-Yen Li e Hung-Kai Ting prevêem qual a posição para onde este pretende deslocar-se. Depois, geram um caminho que leva o utilizador até essa posição. Este caminho evita os obstáculos. Caso pretenda, o utilizador pode rejeitar o caminho que lhe foi sugerido [Li00].

Seth J. Teller e Carlo H. Séquin desenvolveram um método de pré-processamento da visibilidade para modelos arquitectónicos que estejam alinhados com os eixos. Este método recorre a uma organização da cena em células [Tell91].

Brian Salomon, Maxim Garber, Ming C. Lin, e Dinesh Manocha apresentam outra abordagem que gera um caminho livre de colisões entre duas localizações, escolhidas pelo utilizador, num ambiente tridimensional. No entanto, a estratégia criada por estes autores tem a desvantagem de, por vezes, gerar caminhos que são pouco naturais. Além disso, esta

abordagem necessita, previamente, da criação de um mapa. Para modelos tridimensionais com elevado número de polígonos, o tempo necessário para criar o mapa é de várias horas [Salo03].

Marcelo Kallmann, Amaury Aubel, Tolga Abaci, e Daniel Thalmann apresentam em [Kall03] técnicas que permitem controlar personagens humanas em aplicações interactivas.

1.2. Organização da dissertação

A dissertação está organizada nas seguintes secções:

- **Criação da visita guiada**

  Esta secção engloba as subsecções indicadas abaixo. Começa-se por dar uma panorâmica geral acerca de como poderemos criar uma visita guiada através de um modelo complexo.

  - **Detecção automática de células e de portais**

    Nesta secção são abordados os algoritmos que permitem a organização do modelo geométrico em células e portais. As células podem ser comparadas aos vários quartos ou divisões de um apartamento. Os portais permitem a comunicação entre as células.

  - **Selecção das partes do modelo a visitar**

    Aqui são descritos os algoritmos que nos permitem seleccionar quais as células a visitar no modelo. As células a visitar são aquelas que contiverem partes interessantes do modelo. Dentro de cada célula, será, depois, necessário decidir qual a forma de melhor visualizar essas partes interessantes. Ou seja, há que escolher os pontos de vista que transmitem ao utilizador uma maior quantidade de informação [Shan48] acerca das partes interessantes.

  - **Criação do percurso**

    Descrição dos algoritmos que permitem criar um percurso que percorra o modelo. Este percurso deve visitar todas as células que contenham partes interessantes do modelo, permitindo ao utilizador visualizar de forma adequada todas essas partes interessantes.

- **Visita guiada a um museu**

  Descrição detalhada do protótipo que, recorrendo aos algoritmos estudados nas secções anteriores, consegue criar uma visita guiada através de um modelo de pinturas.

- **Conclusão**

  Final da dissertação, com sugestões de ideias para trabalho futuro. Expõem-se também algumas considerações sobre o processo de elaboração da dissertação.

2. Criação de uma visita guiada

Nesta dissertação vão ser apresentados os algoritmos que permitem a criação de uma visita guiada através de um modelo complexo. Sendo assim, um programa que implemente estes algoritmos deverá ser capaz de receber um modelo geométrico arbitrário e retornar um percurso que visite todos os locais de interesse nesse modelo. Na Figura 2.1 encontra-se o percurso de uma visita guiada a um modelo geométrico que representa o primeiro piso do Museu Nacional de Arte Antiga [IPM00]. A visita guiada permite a visualização das obras que são interessantes para um dado utilizador.

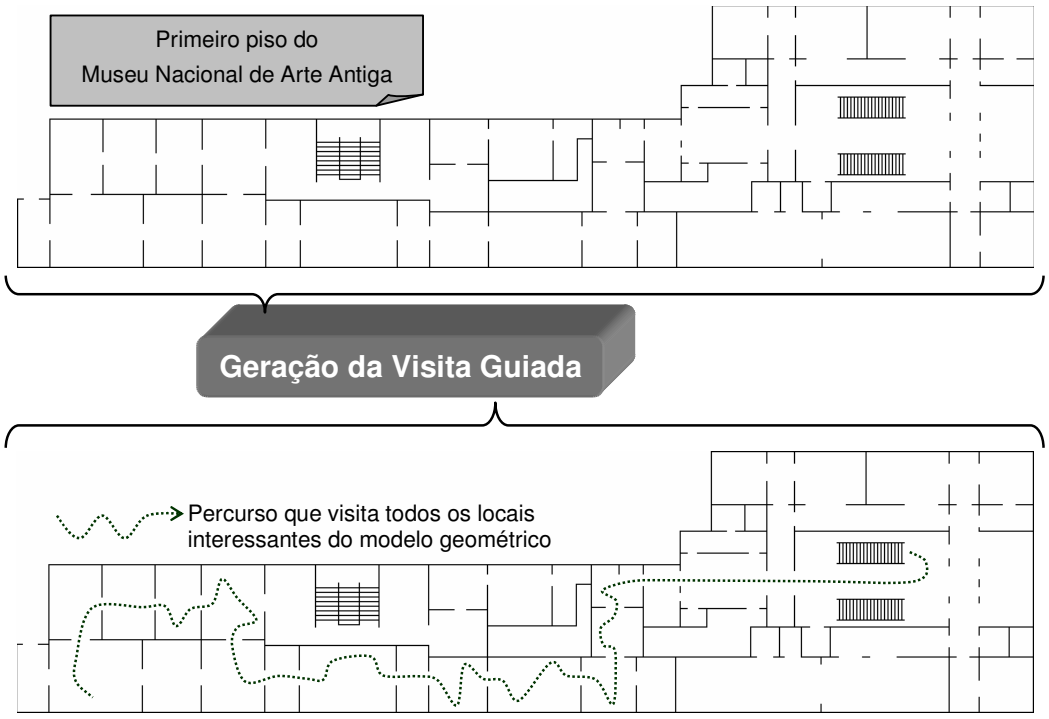


Figura 2.1: Percurso da visita guiada num modelo que representa o primeiro piso do Museu Nacional de Arte Antiga

Dado um determinado modelo, existem diversos percursos que permitem a visualização de todos os locais de interesse. No entanto, alguns percursos podem ser considerados melhores do que outros. Para que um percurso possa ser considerado como bom, deve conformar-se com as seguintes características, que se encontram em [Andú04], e que estão sistematizadas abaixo:

- ✓ **Evita as colisões**
  - ➢ A câmara não pode atravessar os objectos que se encontram no modelo ou colidir com eles.

- ✓ **É interessante**
  - Um bom percurso é interessante, ou seja, mostra ao utilizador as partes mais relevantes do modelo, não perdendo tempo com as partes que não são interessantes. Vamos supor que um utilizador pretende percorrer um modelo geométrico do Museu Nacional de Arte Antiga. Se este utilizador apenas estiver interessado em obras de arte do século XV, então será desnecessário perder tempo com todas as obras de arte que não pertencem a este período. O percurso deverá focar-se apenas naquilo que interessa ao utilizador. Além disso, as partes interessantes do modelo podem ser visualizadas de diversas maneiras. As melhores maneiras são aquelas que transmitem mais informação ao utilizador, devendo o percurso dar-lhes especial importância [Vázq01b, Shan48]. A quantidade de informação que é transmitida ao utilizador terá de ser medida de alguma forma. Mais adiante, nesta dissertação, iremos descrever como isto poderá ser feito.

- ✓ **Não é redundante**
  - Um bom percurso não deve visitar o mesmo local várias vezes. Em muitos modelos, cumprir esta exigência pode ser impossível. Pode haver no modelo, por exemplo, um corredor que tenha de ser constantemente atravessado de forma a alcançar os vários locais de interesse. No entanto, por norma, um bom percurso não repete as visitas ao mesmo local.

- ✓ **Permite que o utilizador não olhe apenas na direcção pela qual avança**
  - De uma forma geral, o utilizador, ao percorrer o modelo, olha em frente, na direcção segundo a qual está a avançar. No entanto, deve ser permitido ao utilizador olhar também noutras direcções. Isto pode ser útil, por exemplo, caso existam objectos no tecto que interesse visualizar.

- ✓ **É ordenado**
  - Um bom percurso não abandona uma área do modelo antes de ter visualizado todos os aspectos importantes dessa área. Uma área do modelo, num edifício, pode ser, por exemplo, um quarto ou um corredor. Sendo assim, o percurso não deve abandonar um quarto, antes de ter mostrado ao utilizador tudo o que de interessante existe para ver nesse quarto.

- ✓ **Ajusta a velocidade**
  - A velocidade com que o utilizador se desloca ao longo do percurso deve abrandar, e até parar, quando ele se aproxima de partes interessantes do modelo. Por outro lado, quando o percurso atravessa zonas em que não existe nada de interessante a visualizar, a velocidade do percurso aumenta.

- ✓ **É suave**
  - Um bom percurso é suave, não existindo mudanças bruscas de velocidade, ou de mudança de direcção.

A geração de uma visita guiada, que cumpra com as características antes enunciadas, tem as seguintes grandes etapas:

1. Detecção automática de células e de portais.
2. Selecção das partes do modelo a visitar.
3. Criação do percurso.

Estas etapas, assim como os algoritmos necessários para as concretizar, irão ser descritas nas próximas secções desta dissertação.

2.1. Detecção automática de células e de portais

A primeira etapa na criação de uma visita guiada através de um modelo consiste na detecção automática de células e de portais. Num edifício, as células são os quartos, as salas, os corredores, etc. Os portais são as fronteiras entre as células, que permitem a comunicação entre estas. Os portais poderão ser, por exemplo, as portas de um edifício.

A organização do modelo em células e portais é útil porque:

- Permite que se analise o modelo região a região, decidindo se uma região deve ser visitada ou não. As várias regiões correspondem às células do modelo [Andú04].

- Facilita a geração de um percurso que evite os obstáculos [Andú04]. Isto porque as células são visitadas uma a uma. Só se abandona uma célula quando todas as áreas de interesse nessa célula foram visualizadas. Sendo assim, na visita a uma determinada célula, o percurso a gerar é restrito à área da célula, em vez de abranger o modelo na sua totalidade.

- A organização do modelo em células e portais pode ainda ser proveitosa para sabermos quais são as áreas do modelo que são visíveis a partir de uma determinada posição. Dessa forma, apenas as células visíveis ou parcialmente visíveis são submetidas ao *rendering*, não se perdendo tempo a processar a restante geometria do modelo [Haum03, Lern03, Lefe03].

A detecção de células e portais não tem obrigatoriamente de ser automática. O próprio criador do modelo poderá, manualmente, efectuar a divisão. No entanto, caso o modelo tenha uma grande dimensão e complexidade essa tarefa é morosa e fastidiosa. Particularmente, nos jogos de computador, em que é comum existirem modelos de enorme dimensão, o tempo e o esforço dispendido nessa tarefa representam um custo acrescido para a empresa, o que poderá tornar o preço de venda ao público do referido jogo mais elevado. Por isso, será vantajoso se existir uma forma de efectuar a detecção de células e portais de forma automática [Haum03, Lefe03].

Para efectuar a detecção de células e portais é necessário [Andú04]:

1. Organizar o modelo em *voxels*.
2. Organizar o modelo em células e detectar os portais.
3. Criar um grafo de células e portais.

Estes três passos irão ser abordados seguidamente.

2.1.1. Organização do modelo em *voxels*

A organização do modelo geométrico em *voxels* é a primeira etapa na detecção automática de células e portais. Primeiro que tudo, é necessário clarificar o termo ***voxel***.

Considerando que R representa o conjunto dos números reais, R^3, por sua vez, representa o espaço tridimensional contínuo. Z é o conjunto dos números inteiros. Z^3 será, por sua vez, o espaço tridimensional discreto [Sá03]. Pode-se visualizar Z^3 como uma grelha de pontos que está contida em R^3. Um ponto na grelha é definido pelas suas coordenadas cartesianas (x, y, z). Um *voxel* ocupa um cubo centrado num dos pontos da grelha (x, y, z), definindo-se como a região contínua (u, v, w) tal que [Huan98, Jone96, Kauf87a, Kauf87b]:

- $x - 0,5 < u \leq x + 0,5$
- $y - 0,5 < v \leq y + 0,5$
- $z - 0,5 < w \leq z + 0,5$

Os *voxels* podem ser vazios ou cheios. Aos *voxels* vazios, ou brancos, é atribuído o valor 0. Aos *voxels* cheios, ou pretos, é atribuído o valor 1. É possível representar um objecto através dos seus *voxels*. Para tal, basta ter uma forma de decidir quais são os *voxels* brancos e quais são os *voxels* pretos. Pode-se considerar que, caso o *voxel* seja intersectado pela geometria de algum dos objectos do modelo, então esse *voxel* é cheio, ou preto. Caso o *voxel* não seja intersectado pela geometria de nenhum dos objectos do modelo, então está-se perante um *voxel* vazio, ou branco [Fole00, Huan98]. Na Figura 2.2 encontram-se dois triângulos aos quais está sobreposta a sua representação em *voxels*. Os *voxels* representados na figura são os cheios.

No entanto, para efeitos da geração automática de células e portais, não basta saber quais são os *voxels* cheios e vazios. Isso é apenas o primeiro passo. Depois, é necessário calcular o **valor distância** (*distance field*).

O valor distância, na Computação Gráfica, pode ser útil para *rendering*, para detecção de colisões, para *morphing* de objectos, para a reconstrução de uma superfície, para a geração de um percurso, etc [Huan01, Jone01c]. No contexto desta dissertação, o valor distância permite a organização do modelo em células e portais, etapa necessária à geração da visita guiada.

A cada *voxel* vai, portanto, ser atribuído um valor distância. O valor distância de um *voxel* vazio é a distância desse *voxel* ao item que estiver mais próximo dele no modelo geométrico. Sendo assim, o valor distância de um *voxel* vazio poderá ser a distância entre esse *voxel* vazio e o *voxel* ocupado que se encontrar mais próximo do *voxel* vazio. Se um *voxel* for cheio, o seu valor distância é 0, uma vez que o *voxel* ocupado mais próximo é ele próprio [Andú04, Jone01b].

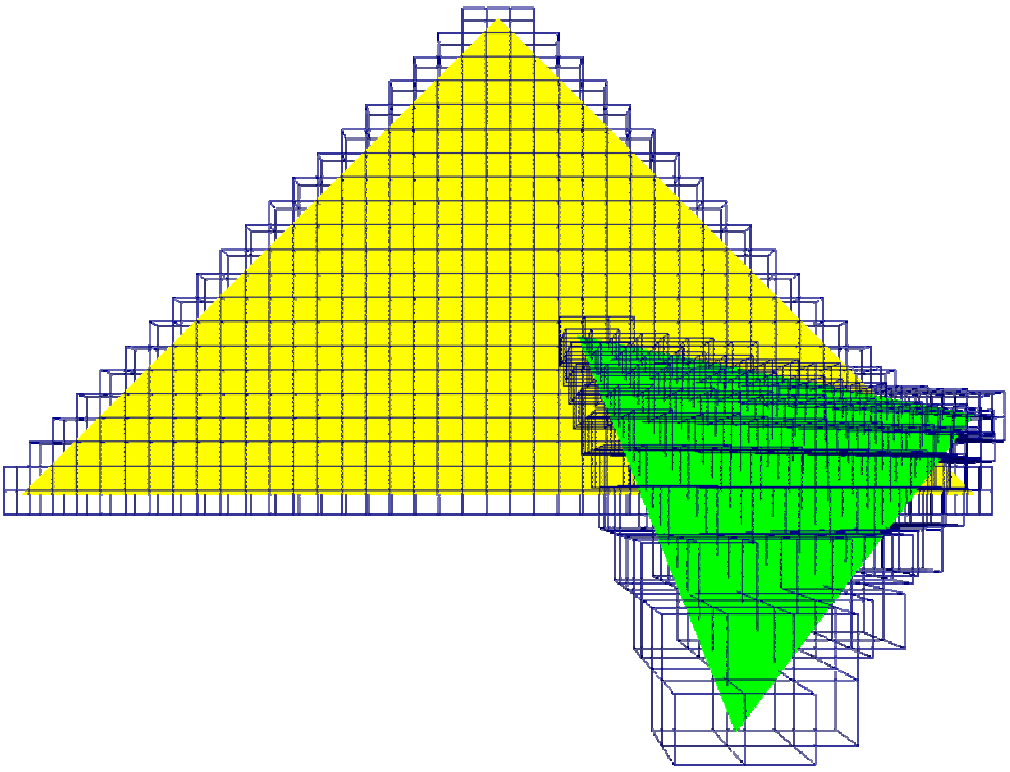


Figura 2.2: Dois triângulos e a sua correspondente representação em *voxels*

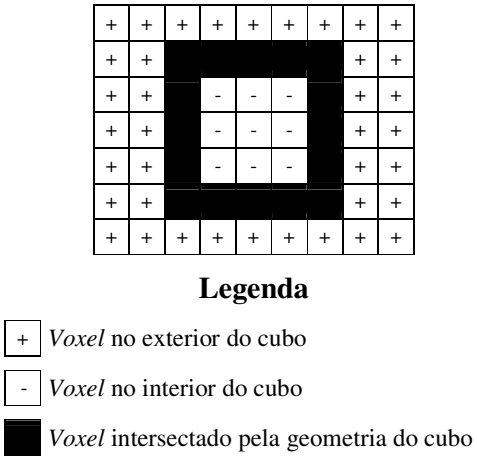


Figura 2.3: *Voxels* no exterior e no interior de um cubo

O valor distância pode ser positivo ou negativo. Se for positivo, é porque o *voxel* está fora do objecto. Se for negativo é porque o *voxel* está dentro do objecto [Jone01b], tal como se pode observar no exemplo da Figura 2.3.

Na Figura 2.3 encontra-se o corte horizontal de uma grelha de *voxels*. Os *voxels* cheios, representados a preto, correspondem ao *voxels* intersectados pela geometria das faces de um cubo. Os *voxels* fora do cubo, com valor distância positivo, são assinalados com o sinal '+'. Os *voxels* dentro do cubo, com valor distância negativo, são assinalados com o sinal '-'.

Se não for relevante saber se o *voxel* está dentro ou fora do objecto, então o valor distância é sempre positivo [Jone01b]. No caso da geração da visita guiada através de um modelo, não existe uma noção clara do que é estar dentro ou fora do objecto, uma vez que existem vários objectos. O que é que pode ser considerado como estando dentro do objecto? É estar no interior de uma célula? É estar no interior de um compartimento? Sendo assim, para a geração da visita guiada, um valor distância apenas positivo é suficiente [Andú04].

Agora que o conceito de valor distância já foi detalhadamente explicado, importa saber como o calcular. Para calcular o valor distância, começa-se por efectuar uma segmentação binária.

- **Segmentação binária (*binary segmentation*)**

  Este método representa a superfície dos objectos do modelo através de uma grelha tridimensional de *voxels*. Para tal, é necessário, de acordo com [Jone01a] e com [Jone01b], começar por recorrer à função de segmentação:

$$f(v) = \begin{cases} 1 & \text{Se o } \textit{voxel} \text{ é intersectado pela superfície do objecto} \\ 0 & \text{Caso contrário} \end{cases} \qquad (2.1)$$

$$\text{Em que } v \in \mathbb{Z}^3$$

  Portanto, de acordo com a função (2.1), é atribuído o valor 1 a todos os *voxels* que forem intersectados por objectos do modelo. Aos *voxels* não intersectados por objectos do modelo é atribuído o valor 0. Esta função faz a diferenciação entre os *voxels* vazios e os cheios, diferenciação essa que já foi anteriormente explicada por outras palavras.

  Caso o modelo seja constituído por uma malha de triângulos, é possível verificar se um triângulo intercepta um *voxel* com recurso ao método apresentado por Thomas Akenine-Moller em [Aken01]. Os *voxels* que forem intersectados por triângulos passam a ser os *voxels* cheios. Os restantes são os vazios.

  Depois de se ter efectuado a segmentação apresentada, procede-se a um **primeiro cálculo do valor distância.** Neste primeiro cálculo do valor distância, todos os *voxels* cheios, ou seja, todos os *voxels* que são intersectados por objectos do modelo, têm valor distância igual a zero. Os *voxels* que não são intersectados por objectos do modelo, ou seja, os *voxels* vazios, ficam, por agora, com o valor distância igual a infinito [Jone01a, Jone01b].

  Depois de se ter realizado a segmentação binária e o primeiro cálculo do valor distância, temos duas opções: o **cálculo simples do valor distância** ou então o **cálculo do valor distância por cortes sucessivos**. Ambas estas abordagens têm vantagens e desvantagens.

  - **Cálculo simples do valor distância**

    O cálculo simples do valor distância, de acordo com [Jone01a] pode ser levado a cabo através do seguinte pseudo-código:

```
Para cada voxel v
   Se o valor distância de v for diferente de 0 então
      Mínimo = infinito
      Para cada voxel c pertencente à grelha de voxels
            Se o valor distância de c for igual a 0 então
                  Distância = Distância Euclidiana entre v e c
                  Se Distância < Mínimo
                        Mínimo = Distância
      O valor distância de v passa a ser igual ao Mínimo
```

Através deste pseudo-código, é medida a distância de cada *voxel* vazio a todos os *voxels* cheios do modelo. O valor distância final de cada *voxel* vazio é a menor das distâncias que for encontrada a um *voxel* cheio. A distância é a distância euclidiana entre os *voxels*.

Na Figura 2.4 encontra-se o resultado do cálculo do valor distância para uma fatia horizontal de uma grelha de *voxels* representativa de um modelo simples. O comprimento da aresta dos *voxels*, no modelo, é de 0.50. A negrito, e com fundo cinzento, estão destacados os *voxels* pretos, correspondentes às paredes, que têm valor distância zero.

1.80	1.41	1.12	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.12	1.41	1.80
1.58	1.12	0.71	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.71	1.12	1.58
1.50	1.00	0.50	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.50	1.00	1.50
1.50	1.00	0.50	**0.00**	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	**0.00**	0.50	1.00	1.50
1.50	1.00	0.50	**0.00**	0.50	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.50	**0.00**	0.50	1.00	1.50
1.50	1.00	0.50	**0.00**	0.50	1.00	1.00	1.00	1.00	1.12	1.41	1.50	1.41	1.12	1.00	1.00	1.00	1.00	0.50	**0.00**	0.50	1.00	1.50
1.50	1.00	0.50	**0.00**	0.50	0.50	0.50	0.50	0.50	0.71	1.12	1.58	1.12	0.71	0.50	0.50	0.50	0.50	0.50	**0.00**	0.50	1.00	1.50
1.50	1.00	0.50	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.50	1.00	1.50	1.00	0.50	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.50	1.00	1.50
1.50	1.00	0.50	**0.00**	0.50	0.50	0.50	0.50	0.50	0.71	1.12	1.58	1.12	0.71	0.50	0.50	0.50	0.50	0.50	**0.00**	0.50	1.00	1.50
1.50	1.00	0.50	**0.00**	0.50	1.00	1.00	1.00	1.00	1.12	1.41	1.80	1.41	1.12	1.00	1.00	1.00	1.00	0.50	**0.00**	0.50	1.00	1.50
1.50	1.00	0.50	**0.00**	0.50	1.00	1.50	1.50	1.50	1.58	1.80	2.12	1.80	1.58	1.50	1.50	1.50	1.00	0.50	**0.00**	0.50	1.00	1.50
1.50	1.00	0.50	**0.00**	0.50	1.00	1.50	2.00	2.00	2.06	2.24	2.50	2.24	2.06	2.00	2.00	1.50	1.00	0.50	**0.00**	0.50	1.00	1.50
1.50	1.00	0.50	**0.00**	0.50	1.00	1.50	2.00	2.50	2.55	2.69	2.92	2.69	2.55	2.50	2.00	1.50	1.00	0.50	**0.00**	0.50	1.00	1.50
1.50	1.00	0.50	**0.00**	0.50	1.00	1.50	2.00	2.50	3.00	3.16	3.35	3.16	3.00	2.50	2.00	1.50	1.00	0.50	**0.00**	0.50	1.00	1.50
1.50	1.00	0.50	**0.00**	0.50	1.00	1.50	2.00	2.50	3.00	3.50	3.81	3.50	3.00	2.50	2.00	1.50	1.00	0.50	**0.00**	0.50	1.00	1.50
1.58	1.12	0.71	0.50	0.71	1.12	1.58	2.06	2.55	3.04	3.54	4.03	3.54	3.04	2.55	2.06	1.58	1.12	0.71	0.50	0.71	1.12	1.58
1.80	1.41	1.12	1.00	1.12	1.41	1.80	2.24	2.69	3.16	3.64	4.12	3.64	3.16	2.69	2.24	1.80	1.41	1.12	1.00	1.12	1.41	1.80
2.12	1.80	1.58	1.50	1.58	1.80	2.12	2.50	2.92	3.35	3.81	4.27	3.81	3.35	2.92	2.50	2.12	1.80	1.58	1.50	1.58	1.80	2.12
2.50	2.24	2.06	2.00	2.06	2.24	2.50	2.83	3.20	3.61	4.03	4.47	4.03	3.61	3.20	2.83	2.50	2.24	2.06	2.00	2.06	2.24	2.50
2.92	2.69	2.55	2.50	2.55	2.69	2.92	3.20	3.54	3.91	4.30	4.72	4.30	3.91	3.54	3.20	2.92	2.69	2.55	2.50	2.55	2.69	2.92
3.35	3.16	3.04	3.00	3.04	3.16	3.35	3.61	3.91	4.24	4.61	5.00	4.61	4.24	3.91	3.61	3.35	3.16	3.04	3.00	3.04	3.16	3.35
3.81	3.64	3.54	3.50	3.54	3.64	3.81	4.03	4.30	4.61	4.95	5.32	4.95	4.61	4.30	4.03	3.81	3.64	3.54	3.50	3.54	3.64	3.81
4.27	4.12	4.03	4.00	4.03	4.12	4.27	4.47	4.72	5.00	5.32	5.66	5.32	5.00	4.72	4.47	4.27	4.12	4.03	4.00	4.03	4.12	4.27

Figura 2.4: Cálculo do valor distância para uma fatia horizontal de uma grelha de *voxels* representativa de um modelo simples

Esta forma de cálculo do valor distância produz resultados precisos. No entanto, caso o modelo seja muito grande, o que resultará num elevado número de *voxels*, então este método pode torna-se moroso e incomportável.

A aplicação deste método ao modelo UNC CThead, com 7 milhões de *voxels*, levou, num computador Athlon de 800 MHz, 2 semanas até terminar, tal como é documentado em [Jone01a].

É possível tornar o cálculo do valor distância mais rápido através da utilização de uma *octree*. Dessa forma, é possível rejeitar partes da *octree* que não contêm *voxels* transversos. Além disso, é também possível rejeitar porções da *octree* que estejam mais longe do que a distância mínima em análise. Mas, ainda recorrendo a uma *octree*, este método tem uma duração superior a duas horas para o modelo UNC CThead, segundo [Jone01b].

- **Cálculo do valor distância por cortes sucessivos**

Caso o modelo contenha muitos *voxels*, o que torna incomportável a utilização do cálculo simples do valor distância, o cálculo do valor distância por cortes sucessivos (*Chamfer Distance Transform*) poderá ser uma boa solução alternativa.

No cálculo do valor distância por cortes sucessivos é efectuada uma propagação local da distância, com recurso à Equação (2.2), e à aplicação de uma matriz de distâncias d_M [Jone01a, Jone01b, Jone01c].

$$D(x,y,z) = \min\left(D\left(x+i, y+j, z+k\right) + d_M\left(i,j,k\right)\right) \qquad \forall i,j,k \in d_M \text{ em que } x,y,z,i,j,k \in \mathbf{Z} \tag{2.2}$$

Na Figura 2.5 encontram-se três exemplos de matrizes de distâncias. Quanto maior o tamanho da matriz utilizada, mais preciso será o cálculo do valor distância. No entanto, o tempo de execução torna-se maior [Grev04, Jone01c].

Tal como Mark W. Jones e Richard Satherley referem, o cálculo do valor distância por cortes sucessivos propaga as distâncias locais através da adição dos valores distância conhecidos dos vizinhos, obtidos através da matriz de distâncias d_M. Cada valor na matriz representa um valor distância local [Jone01a, Jone01b, Jone01c].

De acordo com [Jone01a] e com [Jone01b], a matriz distância é aplicada em duas passagens, da forma que é indicado seguidamente:

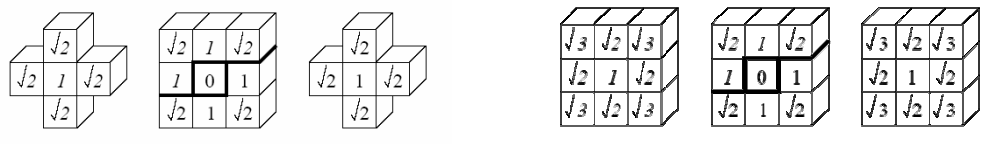
```
/* Primeira passagem */
Para(z=0; z< f_z ; z++)
   Para(y=0; y< f_y ; y++)
         Para(x=0; x< f_x ; x++)
               ValorDistância(x, y, z) = Equação(2.2)

/* Segunda passagem */
Para(z= f_z -1; z≥0; z--)
   Para(y= f_y -1; y≥0; y--)
         Para(x= f_x -1; x≥0; x--)
               ValorDistancia(x, y, z) = Equação(2.2)
```

A primeira passagem utiliza a parte da matriz acima e à esquerda da linha em negrito. A segunda passagem utiliza a parte da matriz abaixo e à direita da linha em negrito [Jone01a, Jone01b]. A linha em negrito pode ser observada na Figura 2.5.

Matriz de distância quase Euclideana de dimensão 3*3*3

Matriz de distância completa de dimensão 3*3*3



Matriz de distância quase Euclideana de dimensão 5*5*5

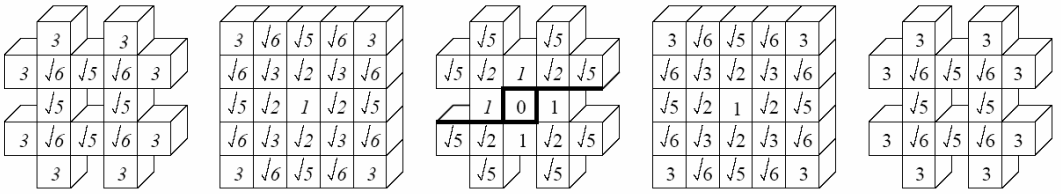


Figura 2.5: Matrizes de distância
Adaptado de: Mark W. Jones e Richard Satherley. Vector-City Vector Distance Transform. Em Computer Vision and Image Understanding, Volume 82, Issue 3, Pages 238-254, June 2001

O cálculo do valor distância por cortes sucessivos tem a vantagem de ser muito rápido, uma vez que cada *voxel* é considerado apenas duas vezes. A desvantagem é que os valores distância calculados são menos precisos do que os obtidos através do cálculo simples do valor distância [Jone01a, Jone01b, Jone01c]. No entanto, de acordo com [Andú04] e com [Haum03] o cálculo do valor distância por cortes sucessivos é suficientemente preciso para a organização do modelo em células e portais.

2.1.2. Organização do modelo em células e detecção de portais

Após a aplicação dos algoritmos abordados na secção anterior, é possível criar uma representação de *voxels* do modelo, sendo atribuído a cada um desses *voxels* um valor distância. O valor distância é necessário à organização do modelo em células, tal como se irá, seguidamente, explicar.

O algoritmo necessário à organização do modelo em células é uma adaptação do algoritmo *watershed transform*, que aqui se designará por algoritmo de segmentação por barragens.

O algoritmo de segmentação por barragens recebe uma superfície e, através de um processo de inundação define, nessa superfície, as bacias (*catchment basins*) e as barragens (*watershed lines*). De início, a superfície está seca, tal como se pode ver na Figura 2.6-1. A água começa, então, a infiltrar-se na superfície, por baixo, a uma velocidade vertical constante. Na Figura

2.6-2 foi detectada a primeira bacia, que é denominada pela letra A. À medida que a água sobe, surgem outros mínimos locais, que dão origem a outras bacias. Na Figura 2.6-3 já existem as bacias A, B, C e D. Se nada for feito, a determinada altura, a água de uma bacia irá fundir-se com a água doutra bacia. Para impedir essa fusão são construídas as barragens. Na Figura 2.6-4 foi construída a primeira barragem, que separa as bacias C e D. A água continua a infiltrar-se, até que toda a superfície esteja submersa. Na Figura 2.6-6 o processo de inundação terminou e apenas as barragens emergem à superfície. Nesta superfície, o algoritmo de segmentação por barragens detectou cinco bacias: a bacia A, a bacia B, a bacia C, a bacia D e a bacia E [Haum03].

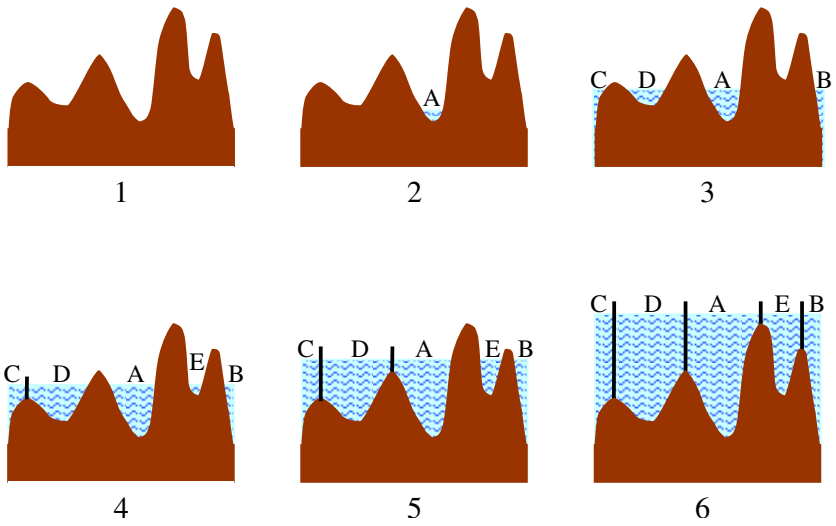


Figura 2.6: Evolução do algoritmo de segmentação por barragens
Adaptado de: D. Haumont, O. Debeir e F. Silicon. Volumetric cell-and-portal generation. Em Eurographics, Volume 22, Number 3, 2003.

O algoritmo de segmentação por barragens, em duas dimensões, é frequentemente utilizado para a segmentação de imagens. Numa imagem em tons de cinzento, as barragens correspondem aos contornos e cada bacia corresponde a um objecto da imagem [Haum03].

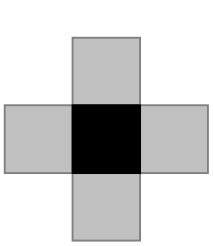
O algoritmo que organiza automaticamente o modelo geométrico em células, modelo esse por onde decorre a visita guiada, tem muitas semelhanças com o algoritmo de segmentação por barragens. No modelo, as bacias corresponderão às células. As barragens são os portais que permitem a comunicação entre as diferentes células.

Para levar a cabo a organização do modelo em células, começa-se por atribuir a todos os *voxels* do modelo um identificador de célula negativo (-1). Os *voxels* com identificador de célula negativo serão, deste modo, os *voxels* que ainda não foram integrados em nenhuma das células.

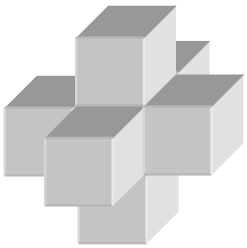
Depois, de entre todos os *voxels*, escolhe-se aquele que tem maior valor distância e atribui-se-lhe um identificador de célula positivo. Esta vai ser a primeira célula do modelo. Pode-se fazer um paralelismo entre este *voxel* e a bacia mais profunda da superfície do algoritmo de segmentação por barragens. Ou seja, quanto maior o valor distância do *voxel*, a maior profundidade este se encontra na bacia.

Seguidamente, é efectuada a propagação do identificador desta nova célula para os *voxels* vizinhos que partilham uma face com o *voxel* em análise. Em duas dimensões os *voxels* vizinhos são quatro. Em três dimensões os *voxels* vizinhos são seis. Na figura Figura 2.7 mostra-se a vizinhança em duas e em três dimensões. Em [Andú04] defende-se que uma

propagação em duas dimensões é suficiente caso a altura da câmara em relação ao solo seja constante durante a visita guiada ao modelo.



(a)
Vizinhança de um *voxel* em duas dimensões.
Voxel em análise a negro e vizinhos a cinzento.



(b)
Vizinhança de um *voxel* em três dimensões.
Voxel em análise no interior.

Figura 2.7: Vizinhança de *voxels* em (a) duas e em (b) três dimensões para efeitos da detecção automática de células e portais

O identificador é propagado do *voxel* em análise para um vizinho se forem cumpridas as seguintes condições [Andú04, Haum03]:

- O identificador do *voxel* em análise tem um valor distância maior ou igual do que o do *voxel* do vizinho.
- O valor distância do vizinho é maior do que zero.

Portanto, a propagação do identificador da célula continua até que toda a célula esteja rodeada por *voxels* que podem ser de três tipos [Andú04, Haum03]:

- *Voxels* com valor distância igual a zero, ou seja, *voxels* ocupados por objectos do modelo.
- *Voxels* com valor distância maior do que os *voxels* da fronteira da célula. Estes *voxels* com valor distância superior irão, posteriormente, ser integrados noutras células.
- *Voxels* que pertencem à fronteira do modelo. Esta situação pode acontecer caso a célula esteja encostada à fronteira do modelo, não havendo nenhuma parede ou objecto que a separe dessa fronteira.

Após se ter propagado o identificador desta primeira célula, escolhe-se, de entre os *voxels* ainda não visitados, o que tem maior valor distância, atribuindo-se-lhe um novo identificador de célula que será novamente propagado [Andú04, Haum03].

O algoritmo termina quando todos os *voxels* vazios pertencerem a uma célula do modelo [Andú04, Haum03].

Na Figura 2.8 encontra-se o resultado da geração automática de células na fatia horizontal de um modelo simples. Todos os *voxels* vazios foram atribuídos a células. Geraram-se seis células identificadas pelos números 0, 1, 2, 3, 4 e 5. Os *voxels* com fundo negro são as paredes, que não foram atribuídas a nenhuma célula e, por isso, têm o valor -1. Os restantes *voxels* exibem o número da célula a que pertencem. Este modelo é o mesmo para o qual foi efectuado o cálculo do valor distância na Figura 2.4.

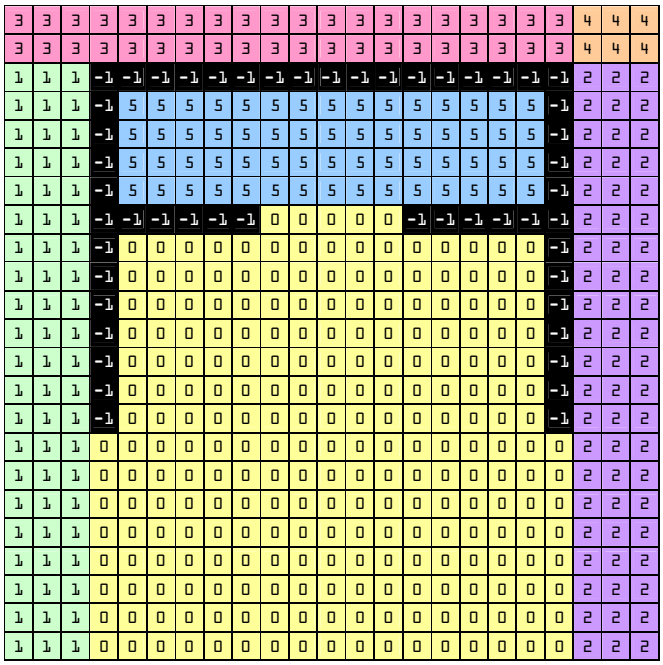


Figura 2.8: Divisão em células de uma fatia horizontal de um modelo simples

Após se ter definido as células do modelo, resta detectar os portais. Caso um *voxel* partilhe uma face com um outro *voxel* pertencente a uma célula diferente, então essa face partilhada fará parte de um portal. Os portais são, deste modo, um conjunto de faces de *voxels* conectadas entre si e partilhadas entre duas células [Andú04]. Na Figura 2.9 estão assinalados, a traço interrompido, os portais entre as células. O modelo utilizado é o mesmo da Figura 2.8. Os *voxels* não foram desenhados de forma a permitir uma melhor visualização dos portais.

O algoritmo consegue, deste modo, efectuar uma organização do modelo em células.

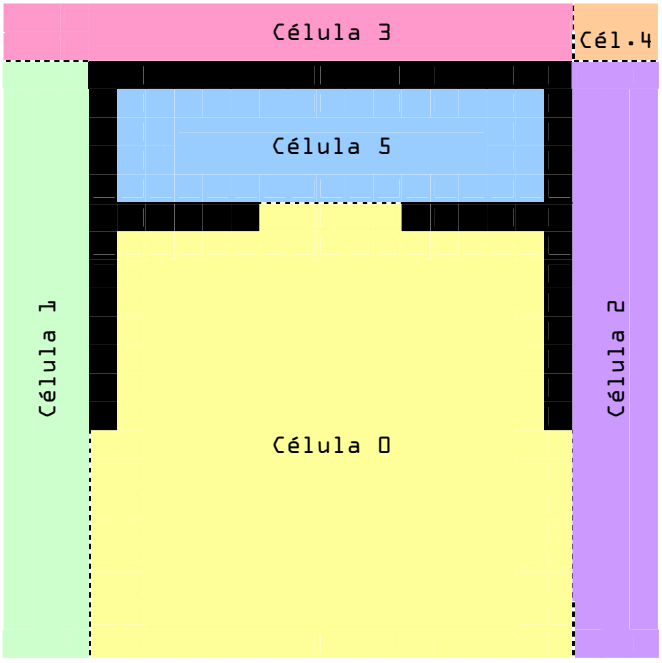


Figura 2.9: Portais numa fatia horizontal de um modelo simples

Existem muitas maneiras de efectuar a divisão de um modelo em células. Por exemplo, um ser humano, podia, de forma manual, escolher a divisão em células e portais que se mostra na Figura 2.10.

O algoritmo apresentado não garante que a divisão em células e portais efectuada seja a melhor possível. Aliás, não existe ainda um consenso acerca do que pode ser considerada a melhor divisão para um modelo arbitrário [Haum03, Lefe03].

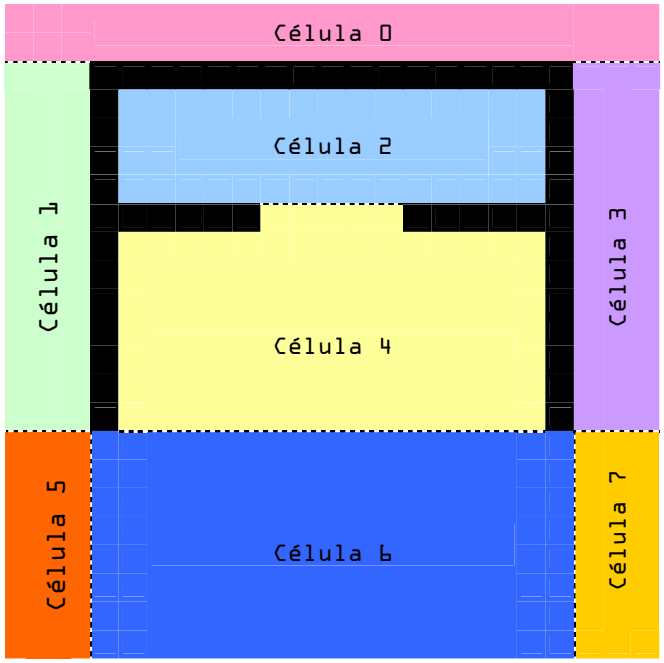


Figura 2.10: Divisão em células e portais efectuada de forma manual.

Há ainda que referir que este algoritmo é muito sensível ao ruído. O modelo simples, utilizado na Figura 2.8 contém apenas paredes, pelo que a divisão em células e portais resultou muito bem. No entanto, se existisse mobília no modelo (armários, cadeiras, etc), a organização em células poderia não ter resultado tão bem. Isto porque a mobília, para efeitos deste algoritmo, é ruído.

O ruído, voltando ao paralelismo com o algoritmo de segmentação por barragens, pode levar à existência de muitas bacias. Muitas bacias implicam muitos máximos locais. Muitos máximos locais implicam a geração de muitas células, células essas eventualmente desnecessárias e apenas causadoras de confusão. Quando existe uma grande profusão de células desnecessárias no modelo, diz-se que estamos perante um problema de sobre-segmentação [Haum03]. As estratégias que permitem resolver ou, pelo menos, atenuar este problema são:

- **Simplificação manual**

  Antes de efectuar a organização do modelo em células, "esvaziar" esse mesmo modelo de toda a mobília (cadeiras, mesas, armários, etc), de forma a conservar apenas as paredes. Após a divisão em células ter sido efectuada, o modelo pode voltar a ser "mobilado" com todos os objectos que, por conveniência, foram removidos. Esta abordagem é simples, mas pode revelar-se fastidiosa caso o modelo seja grande, complexo e contenha uma elevada profusão de mobília [Andú04, Haum03].

- **Fusão ou eliminação de células demasiado pequenas**

  Caso uma célula seja demasiado pequena, essa mesma célula é fundida com uma célula vizinha. A célula escolhida para a fusão é a célula vizinha com a qual a célula demasiado pequena partilha o maior número de faces de *voxels*. Para a utilização desta estratégia será necessário definir, à priori, ou através de uma heurística, o que se considera como sendo uma célula demasiado pequena [Andú04, Haum03].

  No que se refere ao modelo simples apresentado na Figura 2.8, podia-se estabelecer que todas as células devem ter, pelo menos, nove *voxels* de área. Sendo assim, a célula 4, com uma área de apenas seis *voxels*, seria considerada demasiado pequena. A célula 4 tem como vizinhas a célula 3 e a célula 2. A célula escolhida para a fusão seria a célula 2, uma vez que é com ela que a célula 4 partilha o maior número de faces de *voxels*. A célula 4 e a célula 2 passariam, deste modo, a ser consideradas como uma única célula.

  Podia acontecer que a célula considerada como sendo demasiado pequena não tivesse vizinhas. Nesse caso, C. Andújar, P. Vazquez e M. Fairén recomendam que essa célula seja eliminada [Andú04]. Na Figura 2.11 encontra-se um exemplo dessa situação. A célula 2 tem uma área menor do que o limite previamente estabelecido de nove *voxels*. Esta célula não tem vizinhas porque está totalmente rodeada por *voxels* ocupados pela geometria do modelo. Recorda-se que os *voxels* ocupados pela geometria do modelo, tais como paredes, são identificados pelo número -1. Sendo assim, os *voxels* da célula 2 passam também a ser considerados como inacessíveis, sendo esta célula eliminada.

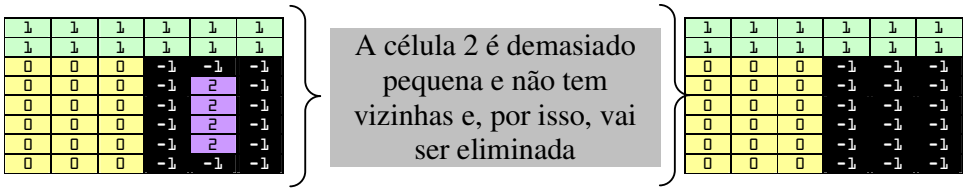


Figura 2.11: Eliminação de células demasiado pequenas e sem vizinhas

- **Fusão de células que partilham um elevado número de faces de *voxels***

  Se duas células partilharem um número de faces de *voxels* que seja considerado demasiado grande, essas células são fundidas numa única célula. Os *voxels* cujas faces são partilhadas por duas células correspondem aos portais [Andú04].

  À semelhança do que acontece na estratégia anterior, também aqui é necessário definir, à priori, ou através de uma heurística, o que se considera como sendo um número demasiado grande de faces de *voxels* partilhadas por duas células. Por outras palavras, é necessário definir o que se considera como sendo um portal demasiado grande.

  No caso do modelo simples apresentado na Figura 2.8, podia-se estabelecer que os portais não poderiam ter mais do que 6 *voxels* de extensão. Uma vez que o portal entre a célula 0 e a célula 1 tem 8 *voxels* de extensão, estas duas células seriam fundidas numa só. A seguir, seria efectuada nova fusão com a célula 2, uma vez que a extensão do portal também é maior do que o limite estabelecido.

- **Propagação do identificador das células com tolerância decrescente**

  C. Andújar, P. Vazquez e M. Fairén sugerem que a propagação do identificador da célula de um *voxel* v, para um *voxel* v', seu vizinho, se faça apenas caso:

$$0 < D_{v'} \leq D_v + \varepsilon \tag{2.3}$$

  ε é um valor de tolerância que decresce para 0 à medida que a célula aumenta. Através da aplicação desta tolerância, pequenas alterações do valor distância dos *voxels* perto do *voxel* de origem da célula terão um menor impacto na propagação do identificador da célula [Andú04].

No que se refere às estratégias apresentadas para evitar a sobre-segmentação, pode-se considerar que a primeira, a simplificação manual, proporciona bastante precisão, mas é potencialmente fastidiosa num modelo de larga dimensão. As três restantes estratégias são de carácter mais automático. Podem ser utilizadas em conjunto ou individualmente. Em certos modelos, poderá ser necessário experimentá-las numa lógica de tentativa e erro até se obter uma organização em células que seja considerada conveniente. Aliás, como já se referiu anteriormente, existem muitas formas de dividir um modelo em células, não existindo ainda um consenso acerca do que é a melhor divisão possível [Haum03, Lefe03].

2.1.3. Criação do grafo de células e portais

Após se ter organizado o modelo em células e portais, é necessário criar o grafo de células e portais. No grafo de células e portais, os vértices são as células. Os ramos que ligam os nós são os portais [Andú04, Haum03, Lefe03].

Na Figura 2.12 encontra-se o grafo de células e portais para o modelo simples apresentado na Figura 2.8.

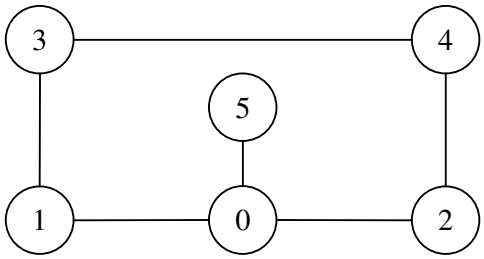


Figura 2.12: Grafo de células e portais para um modelo simples

O grafo de células e portais irá, posteriormente, ser útil na construção do percurso que visita o modelo. Apenas serão visitadas as células que contêm partes interessantes do modelo.

2.2. Selecção das partes do modelo a visitar

Após o modelo estar organizado em células e portais é necessário construir um percurso que permita ao utilizador ficar com uma boa ideia desse modelo. Esse percurso é gerado pelo computador, com o mínimo de esforço e intervenção por parte do utilizador.

Tal como já referimos anteriormente, o percurso gerado deve ser interessante. Um percurso interessante é aquele que conduz o utilizador a posições com vistas de qualidade. Uma **vista de qualidade** é aquela em que se visualizam partes interessantes do modelo, sendo essas partes interessantes visualizadas da melhor forma possível. Sendo assim, como fazer para decidir quais são as posições que permitem obter vistas de qualidade?

Esta decisão pode ser manual. Alguém que conheça bem o modelo poderia escolher as posições a partir das quais esse modelo seria visualizado. No entanto, nem sempre é possível encontrar alguém com esse conhecimento. Além disso, essa escolha manual, em modelos de grande dimensão, é uma tarefa morosa e enfadonha para quem tenha de realizá-la. Por isso, é vantajoso se o computador puder escolher, sozinho, quais as posições que fornecem as vistas de melhor qualidade.

Uma vista de qualidade mostra um objecto da melhor forma possível. Ora, o que é a melhor forma possível de visualizar um objecto?

Supondo que o objecto a visualizar é um cubo, uma vista de qualidade deverá permitir ao utilizador aperceber-se de que está perante um cubo. Na Figura 2.13 encontram-se dois cubos. Na vista da esquerda, o cubo é visualizado de frente, pelo que o utilizador poderia ficar na dúvida se estaria perante um cubo ou perante um quadrado. Na vista da direita é compreensível que se está perante um cubo. A vista da direita tem, portanto, maior qualidade do que a da esquerda.

(a) Má vista de um cubo (b) Boa vista de um cubo

Figura 2.13: Má vista e boa vista de um cubo

Na Figura 2.14 encontra-se outro exemplo, desta vez, com um objecto constituído por dois paralelepípedos, um maior e outro mais pequeno. Na vista da esquerda o utilizador não se apercebe da existência do segundo paralelepípedo mais pequeno, que ficou escondido atrás do paralelepípedo maior. Na vista da direita, é perceptível que o objecto é constituído por dois paralelepípedos.

(a) Má vista de um objecto (b) Boa vista de um objecto

Figura 2.14: (a) Boa vista e (b) má vista de um objecto constituído por dois paralelepípedos

Não existe ainda, em termos matemáticos, um consenso acerco do que é uma vista de qualidade. Pode-se considerar que uma vista de qualidade é aquela que transmite a maior quantidade de informação possível ao utilizador acerca do objecto visualizado, permitindo-lhe identificar correctamente esse mesmo objecto. A quantidade de informação transmitida ao utilizador irá ser avaliada com base na geometria do objecto [Vázq03a].

A qualidade de uma vista pode ser avaliada de acordo com os seguintes critérios:

- **Número de faces visíveis**

  De acordo com este critério, a melhor vista é aquela em que é visível o maior número de faces [Soko05, Vázq03a, Vázq01a, Vázq03b]. Este é um critério que dá grande importância aos detalhes, uma vez que as zonas com maior concentração de faces são aquelas que recebem mais atenção. Na Figura 2.15 (a) encontra-se uma vista de um cubo em que o número de faces visíveis foi maximizado. Todas as faces do cubo são compostas por um triângulo verde e por um triângulo branco, à excepção de uma, que é composta por quatro triângulos verdes e por quatro triângulos brancos. Repare-se que a vista é de qualidade, porque permite ao utilizador perceber que está perante um cubo. Além disso, o pormenor da face de oito triângulos é visível na vista.

  Na Figura 2.15 (e) encontra-se a melhor vista de um coelho em que o número de faces visíveis foi maximizado. O modelo geométrico do coelho está disponível, em formato PLY, em [Stan05] e em [GIT06]. Nesta vista, o coelho é perfeitamente perceptível.

  No entanto, nem sempre a vista com o maior número de faces permite ao utilizador identificar correctamente o objecto. A Figura 2.15 (c) é a melhor vista de uma garrafa de secção quadrada, de acordo com o critério de maximização do número de faces visíveis. A garrafa é colorida com cinco cores diferentes. Nesta vista, o utilizador poderia aperceber-se de que existe uma abertura no cimo do objecto. Porém, a altura do objecto não é perceptível e, portanto, seria difícil compreender que a imagem representa uma garrafa.

- **Área projectada total**

  Segundo este critério, a melhor vista é aquela em que área projectada for maior. A área projectada total é um bom indicador de quanta informação existe na cena [Soko05, Vázq03a, Vázq01a, Vázq03b].

  Nas Figura 2.15 (b) e na Figura 2.15 (f) a área projectada total foi maximizada. Repare-se que, tanto no caso do cubo, como no caso do coelho os resultados não são os melhores.

Na Figura 2.15 (b) a face mais detalhada do cubo não aparece. Além disso, é muito mais fácil reconhecer que se está perante um cubo na Figura 2.15 (a), do que na Figura 2.15 (b). Portanto, no caso da visualização do cubo, o critério de maximização da área projectada comporta-se pior do que o critério de maximização do número de faces visíveis.

Na Figura 2.15 (f) é visível que estamos perante um coelho. No entanto, a vista da Figura 2.15 (e) permite reconhecer o coelho mais facilmente. Portanto, no caso do coelho, o critério da maximização do número de faces visíveis também é melhor.

A Figura 2.15 (d) maximiza a área projectada da garrafa. Esta vista tem a vantagem de mostrar a altura da garrafa. Contudo, a abertura no topo da garrafa não é visível. Sendo assim, no que se refere à garrafa, nenhum destes critérios se comporta bem. A maximização do número de faces visíveis não deu importância ao comprimento da garrafa. E a maximização da área projectada deixou escapar a abertura.

- **Minimização das faces degeneradas**

  Este critério tenta minimizar o número de faces degeneradas. Uma face é degenerada quando o ângulo entre a normal da face e a direcção para onde a câmara está apontada é de $\pi/2$ radianos. A melhor vista é aquela em que existir um menor número de faces degeneradas. No entanto, este critério não assegura que se consiga ver uma grande quantidade de pormenores na vista escolhida. O método não prevê também a forma de lidar com possíveis oclusões [Vázq03a].

Portanto, como é possível ver, todos estes critérios têm fraquezas. Tanto a maximização do número de faces visíveis, como a maximização da área projectada, por vezes, não permitem que o utilizador identifique o objecto representado. A minimização das faces degeneradas, por seu turno, não contempla o problema das oclusões.

Seguidamente, iremos apresentar outros critérios que conseguem ultrapassar estas fraquezas. Um deles é a entropia dos pontos de vista. Outro é a Qualidade Kullback-Liebler. Um outro baseia-se na curvatura da superfície dos objectos. Todos estes critérios avaliam a qualidade de uma vista.

No entanto, primeiro, aborda-se um pouco da Teoria Matemática da Informação, uma vez que os critérios que iremos apresentar têm com ela algumas ligações.

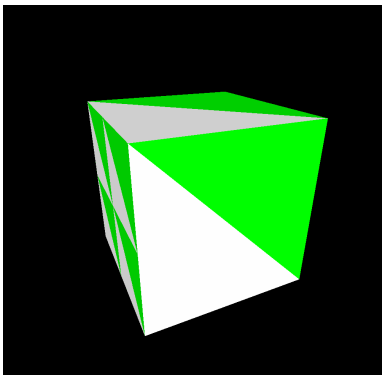
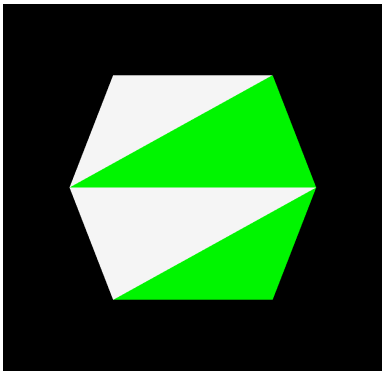
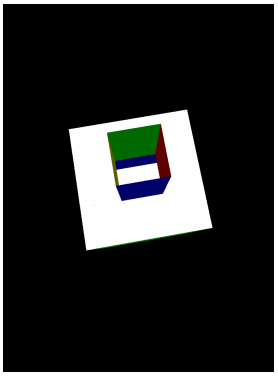
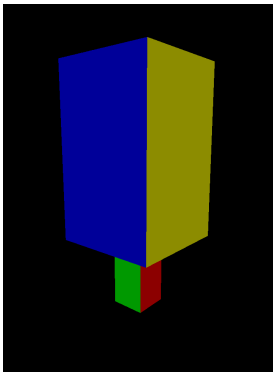
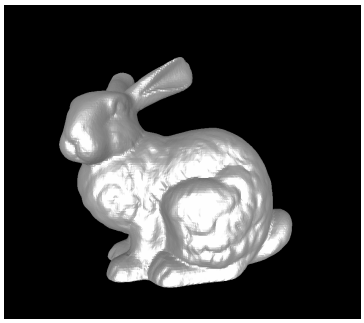
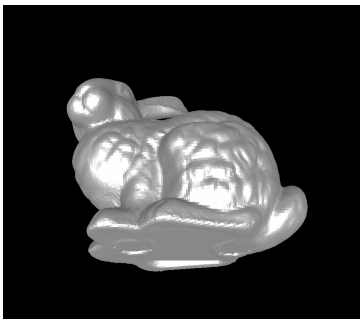
Melhor vista de vários objectos	
Maximização do número de faces visíveis	Maximização da área projectada total
(a) 	(b) 
(c) 	(d) 
(e) 	(f) 

Figura 2.15: Boas vistas de um cubo, uma garrafa e um coelho, de acordo com o critério do número de faces visíveis e de acordo com o critério da área projectada total

2.2.1. A Teoria Matemática da Informação

A Teoria Matemática da Informação foi criada por Claude Shannon e apresentada em 1948. De acordo com esta teoria a informação está fortemente relacionada com a incerteza. Quanto maior a incerteza, maior a informação. Vamos supor que um emissor pretende comunicar uma mensagem a um receptor. Se o receptor souber, à partida, qual é a mensagem que o receptor lhe vai entregar, então a informação contida nessa mensagem é nula, não havendo qualquer utilidade em transmiti-la através do canal de comunicação. Só há troca de informação se existirem várias mensagens possíveis, não sabendo o receptor qual delas é que o emissor lhe irá entregar [Shan48].

A Teoria Matemática da Comunicação associa cada uma das mensagens possíveis com uma probabilidade. A probabilidade de cada uma das mensagens é do conhecimento do receptor. A quantidade de informação dependerá da probabilidade das mensagens. Se uma das mensagens tiver uma elevada probabilidade, em relação às restantes, a quantidade de informação diminui, uma vez que o receptor tem um bom palpite acerca de qual das mensagens irá receber [Shan48].

Quando a probabilidade de todos os eventos é igual, a quantidade de informação comunicada ao receptor é máxima. Isto porque o receptor não pode fazer nenhuma ideia acerca de qual será a mensagem que é mais provável receber [Shan48, Mac03].

A informação pode ser medida com recurso à Equação (2.4) [Shan48, Mac03].

$$H = -\sum_{i=1}^{n} p_i \log p_i \tag{2.4}$$

H é a quantidade de informação e $p_1, p_2, \ldots, p_n$ são as probabilidades das mensagens associadas a uma dada situação. A informação é medida em bits, pelo que o logaritmo é na base 2. Segundo Shannon, de acordo com esta fórmula, tem-se que [Shan48, Mac03]:

- $H = 0$ se e apenas se todos os p_i forem iguais a 0 (zero), excepto um deles que é igual a 1. Neste caso, o receptor já sabe qual é a mensagem que vai receber, e portanto a quantidade informação é zero.
- *H* é máximo quando todos os p_i forem iguais, ou seja, quando $p_i = 1/n$, em que *n* é o número total de mensagens. Esta é situação de maior incerteza, em que o receptor não sabe qual é a mensagem que é mais provável receber, uma vez que todas têm igual probabilidade.
- *H* é sempre maior ou igual a zero.

A Entropia Relativa, ou Distância Kullback-Leibler entre duas distribuições de probabilidades $p = \{p_i\}$ e $q = \{q_i\}$, para uma variável aleatória X, é a indicada na Equação (2.5) [Mac03, Vázq03a].

$$D_{KL}(p \parallel q) = \sum_{i=1}^{n} p_i \log \frac{p_i}{q_i} \tag{2.5}$$

Na Equação (2.5) , para efeitos de continuidade, utilizam-se as seguintes convenções [Mac03, Vázq03a]:

- $0 \log 0 = 0$
- $p_i \log \frac{p_i}{0} = \infty$, se $p_i > 0$
- $0 \log \frac{0}{0} = 0$

2.2.2. Avaliação da qualidade de uma vista

Nesta secção apresentam-se várias estratégias para avaliar a qualidade de um ponto de vista, e que são as seguintes:

- Entropia dos pontos de vista.
- Qualidade Kullback-Leibler.
- Qualidade baseada na curvatura das superfícies.

Relembramos que um ponto de vista com qualidade é aquele que apresenta ao utilizador vistas de qualidade. Uma vista de qualidade permite ao utilizador identificar correctamente o objecto ou os objectos presentes nessa vista.

2.2.2.1. Entropia dos pontos de vista

A entropia dos pontos de vista está fortemente relacionada com a Teoria Matemática da Informação. A entropia dos pontos de vista mede a qualidade de um ponto de vista. Sendo assim, em analogia com a teoria de Claude Shannon, um bom ponto de vista é aquele que transmite muita informação geométrica ao utilizador.

Para que a entropia dos pontos de vista possa ser calculada, segundo [Vázq03a], há que ter em conta os seguintes requisitos:

- A geometria do modelo é conhecida.
- Os polígonos do modelo são planos.
- Cada polígono é considerado como sendo uma face pertencente a um objecto do modelo.
- Só a informação geométrica do modelo é relevante para o cálculo da entropia dos pontos de vista. A cor dos objectos do modelo não é relevante para esse cálculo.

A entropia dos pontos de vista é calculada da forma que é indicada na Equação (2.6) [Sber02, Plem04, Vázq03a, Vázq02a, Vázq02b].

$$I(C,p) = -\sum_{i=0}^{N_f} \frac{A_i}{A_t} \log \frac{A_i}{A_t} \tag{2.6}$$

I é a entropia dos pontos de vista. A entropia é calculada para uma cena C e para um ponto p, nessa cena.

N_f é o número de faces da cena.

A_i é a área projectada da face i numa esfera centrada no ponto de vista p.

A_0, particularmente, é a área projectada do fundo em cenas abertas.

A_t é a área total da esfera. Numa cena fechada, ou caso do ponto p não se consiga ver o fundo, então a esfera é totalmente coberta pela faces projectadas, o que implica que $A_0 = 0$.

Portanto, A_i / A_t é a visibilidade da face i no ponto p. Quanto menor for a distância entre a face i e o ponto de vista, maior é a visibilidade dessa face. Quanto melhor o ângulo através do qual a face é observada, maior é a visibilidade dessa face [Sber02, Vázq03a, Vázq02a, Vázq02b].

Na Figura 2.16 pode observar-se a projecção de algumas faces na esfera centrada no ponto de vista p [Sber02, Vázq02a, Vázq02b].

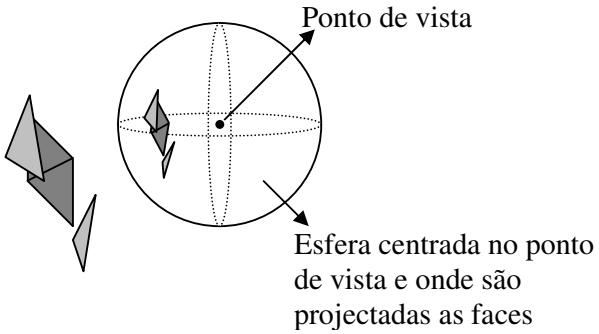


Figura 2.16: Projecção de faces na esfera centrada num ponto, com vista ao cálculo da entropia da vista desse ponto

A entropia de Claude Shannon serve de base à entropia dos pontos de vista, tal como se pode ver no Quadro 2.1 [Sber02].

Entropia de Shannon	Entropia dos pontos de vista
Recorre à probabilidade das mensagens $p_1, p_2, \ldots, p_n$	Recorre à visibilidade das faces em relação a um ponto *p*. A visibilidade de uma face é calculada da seguinte forma: $\text{Visibilidade da face } i = \frac{A_i}{A_t}$ A_i = Área projectada da face *i* sobre a esfera centrada no ponto de vista A_t = Área total da esfera
A soma das probabilidades das mensagens é igual a um.	A soma das visibilidades das faces é igual a um.
A informação atinge o seu máximo se as probabilidades das mensagens são iguais, ou seja, quando qualquer que seja p_i : $p_i = \frac{1}{n}$	A informação visível de um ponto é máxima se, nesse ponto, todas as faces tiverem igual área projectada na esfera, centrada no ponto, ou seja, quando qualquer que seja a face *i*: $\text{Visibilidade da face } i = \frac{A_t / N_f}{A_t} = \frac{1}{N_f}$
$H = -\sum_{i=1}^{n} p_i \log p_i$	$I(C, p) = -\sum_{i=0}^{N_f} \frac{A_i}{A_t} \log \frac{A_i}{A_t}$

Quadro 2.1: Relação entre a Entropia de Shannon e a Entropia dos Pontos de Vista

É importante que a área projectada do fundo seja utilizada na fórmula da entropia dos pontos de vista por diversas razões [Sber02, Vázq03a, Vázq02b]:

- Se não se incluir a área projectada do fundo, a soma da visibilidade das faces não será 1. Ora, de acordo com a entropia de Shannon a soma das probabilidades das várias mensagens deve ser 1. Por isso, a soma da visibilidade das faces também deverá ser 1, ou então a medida da entropia não será consistente.
- O fundo ajuda o utilizador a compreender melhor a cena ou o objecto em visualização, tal como se pode verificar na Figura 2.17. Na Figura 2.17 (a), o fundo não é visível, e por isso é mais difícil para o utilizador saber perante que objecto está. Na Figura 2.17 (b) o fundo é visível, o que permite ao utilizador aperceber-se que está perante um cubo. Além disso, na vista da direita, o utilizador compreende, também, que o cubo está isolado na cena.

(a) (b)

Figura 2.17: Duas vistas do mesmo objecto, (a) sem recurso ao fundo e (b) com recurso ao fundo

Segundo [Vázq03a] o **cálculo da entropia de um ponto de vista** pode ser efectuado de duas formas:

- Calcular a área projectada de cada uma das faces sobre a superfície da esfera cujo centro é o ponto de vista, com recurso a fórmulas matemáticas, e tendo em conta as oclusões que podem surgir entre as faces.
- Recorrendo às funcionalidades de um software que permita o interface com o hardware gráfico, tal como o OpenGL [Shre04]. Para tal, procede-se da seguinte maneira [Barr00, Klei00, Sber02, Vázq02a, Vázq02b]:
  1. Colorir todas as faces existentes na cena com cores diferentes. Dessa forma, será possível identificar cada uma das faces numa projecção.
  2. Seleccionar o ponto de vista para o qual se pretende calcular a entropia.
  3. Obter as seis projecções que envolvem o ponto de vista, tal como se pode observar na Figura 2.18 (o automóvel e as duas bicicletas visíveis na Figura 2.18 pertencem à galeria de objectos do CorelDREAM 3D 8.0 [Core97]).
  4. Ler o *buffer* de cada uma das seis projecções de forma a calcular a visibilidade de cada face. Para calcular a visibilidade de uma face somam-se todos os *pixels* dessa face que foram encontrados nas seis projecções. Lembra-se que cada face tem uma cor que é única. Para que o cálculo seja mais preciso, pode multiplicar-se cada *pixel* pelo seu ângulo sólido.
  5. Após se ter calculado a visibilidade de cada face, aplica-se a Equação (2.6), de forma a calcular a entropia do ponto de vista.

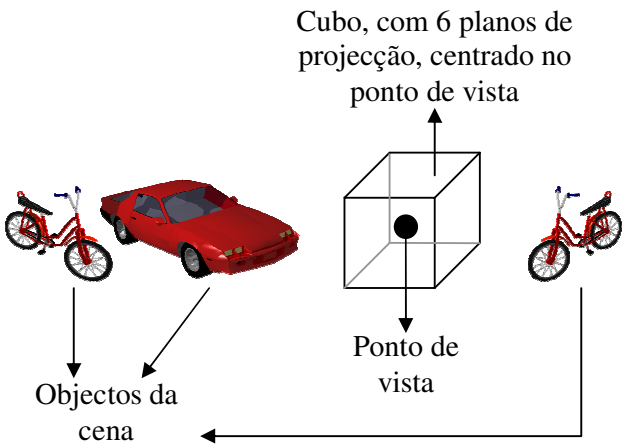


Figura 2.18: Seis projecções que envolvem um ponto de vista

A segunda opção, que recorre às funcionalidades do OpenGL é, sem dúvida, a mais simples de implementar. Além disso, é rápida e tem em conta, para o cálculo da entropia, apenas as faces que projectem, pelo menos, 1 *pixel* nas seis projecções que envolvem o ponto de vista. Portanto, recorrendo ao OpenGL, é mais fácil medir apenas a entropia visível [Vázq03a].

Caso o objecto esteja totalmente contido no *frustum* da câmara, tal como se mostra na Figura 2.19, será possível acelerar a velocidade de execução processando apenas o conteúdo de uma das seis projecções do ponto de vista [Sber02, Vázq02b].

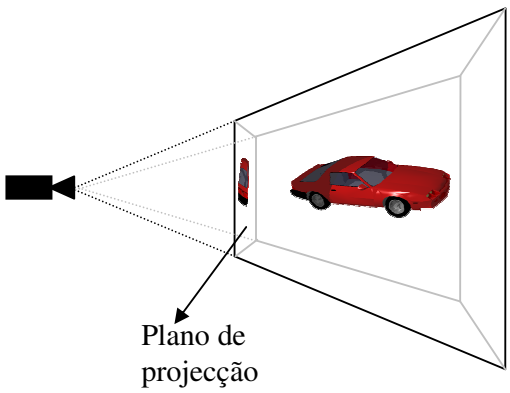


Figura 2.19: Objecto totalmente dentro do *frustum* da câmara

Vistas com entropia mais elevada de vários objectos

(a) (b) (c)

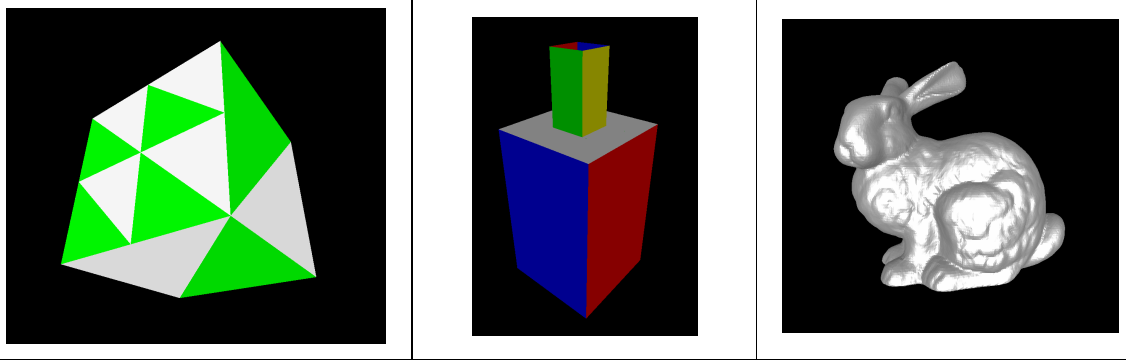


Figura 2.20: Vistas com entropia mais elevada de um cubo, de uma garrafa e de um coelho

Na Figura 2.20 encontram-se as vistas com entropia mais elevada de um cubo, de uma garrafa e de um coelho.

A melhor vista, para estes mesmos objectos, de acordo com o critério de maximização do número de faces visíveis, e de acordo com o critério de maximização da área projectada total já foi anteriormente apresentada na Figura 2.15.

No caso do cubo, na Figura 2.20 (a) é dado mais destaque à face dividida em oito triângulos. Na Figura 2.15 (a), a face mais detalhada também se vê, mas não se vê tão bem.

Na garrafa, que se pode visualizar na Figura 2.20 (b), a entropia dos pontos de vista mostra tanto a altura do objecto, como a abertura no cimo da garrafa. A vista da garrafa apresentada na Figura 2.15 (c), e escolhida de acordo com o critério de maximização do número de faces visíveis, não mostrava o comprimento da garrafa. A Figura 2.15 (d), que foi escolhida de acordo com o critério de maximização da área projectada, deixou escapar a abertura da garrafa. Portanto, no que se refere à garrafa, a entropia dos pontos de vista consegue conjugar o melhor de dois mundos, mostrado tanto o comprimento da garrafa, como a sua abertura.

Finalmente, no que se refere ao coelho, a vista escolhida pela entropia dos pontos de vista (Figura 2.20 (c)) e a vista escolhida pela maximização do número de faces visíveis (Figura 2.15 (e)) é a mesma. A maximização da área projectada, como já foi antes demonstrado na Figura 2.15 (f), comporta-se um pouco pior.

2.2.2.2. Qualidade Kullback-Leibler

Mateu Sbert, Dimitri Plemenos, Miquel Feixas e Francisco González medem a qualidade de um ponto de vista com recurso à Distância Kullback-Leibler, já anteriormente apresentada na secção 2.2.1 desta dissertação. Para tal, propõem a utilização da Equação (2.7) [Sber05].

$$KL(V) = \sum_{i=1}^{N_f} \frac{a_i}{a_t} \log \frac{\frac{a_i}{a_t}}{\frac{A_i}{A_T}} \tag{2.7}$$

V é o ponto de vista para o qual é avaliada a qualidade, baseada na Distância Kullback-Leibler. Deste modo, a Equação (2.7) permite medir a Qualidade Kullback-Leibler de um ponto.

N_f é o número de faces da cena.

a_i é a área projectada da face i na esfera centrada no ponto de vista V. Tal como na Entropia dos pontos de vista, pode-se colorir cada face da cena com uma cor diferente. Dessa forma, consegue-se, depois, identificar cada uma das faces na projecção.

a_t é calculado com recurso à Equação (2.8). Tal como se pode comprovar pela Equação (2.8), ao contrário da Entropia dos pontos de vista, a Distância Kullback-Leibler não recorre ao uso do fundo.

A_i é a área real da face *i*.

A_T é a área total da cena, que pode ser calculada com recurso à Equação (2.9).

$$a_t = \sum_{i=1}^{N_f} a_i \tag{2.8}$$

$$A_T = \sum_{i=1}^{N_f} A_i \tag{2.9}$$

Repare-se que a Qualidade Kullback-Leibler é efectivamente baseada na Distância Kullback-Leibler, tal como se pode verificar na Figura 2.21. A distribuição de probabilidades *p* é substituída pela área relativa das faces projectadas sobre a esfera cujo centro é o ponto de vista *V*. A distribuição de probabilidades *q* é substituída pela área relativa das faces [Sber05].

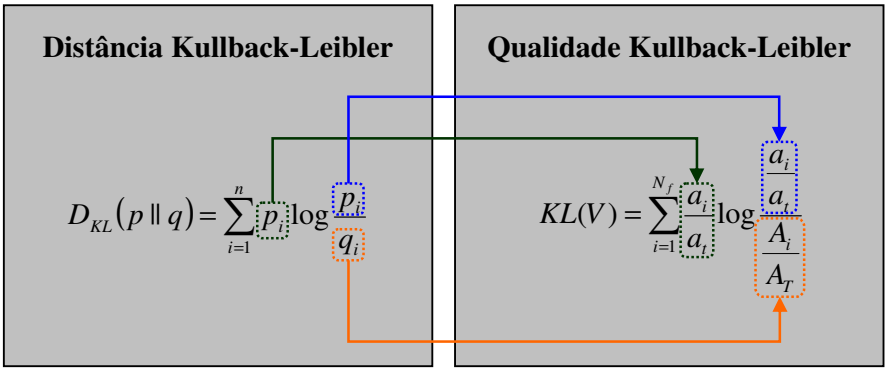


Figura 2.21: Relação entre a Distância Kullback-Leibler e a Qualidade Kullback-Leibler

A Qualidade Kullback-Liebler pode ser interpretada como a distância entre a distribuição normalizada das faces projectadas e a distribuição normalizada das áreas reais das faces. Quando a distribuição das faces projectadas é igual à distribuição das áreas reais das faces,

$KL(V)$ atinge o seu valor mínimo, ou seja, zero. Esta é a situação ideal. Isto significa que, quanto menor for o valor devolvido por $KL(V)$, maior é a qualidade do ponto de vista [Sber05]. É de salientar que, na Entropia dos pontos de vista, a lógica é inversa, porque quanto maior for o valor da entropia, maior é a qualidade do ponto de vista.

A Qualidade Kullback-Liebler é menos sensível do que a Entropia dos pontos de vista à elevada concentração de faces numa zona da cena. A Entropia dos pontos de vista tem tendência para atrair as zonas com elevada concentração de faces. A Qualidade Kullback-Liebler privilegia uma visão balanceada do objecto ou cena [Sber05].

Em favor desta visão balanceada, a Qualidade Kullback-Liebler poderá até abdicar da visualização de algumas faces da cena, tal como se pode observar na Figura 2.22 em que se mostra o melhor ponto de vista de um cubo, à esquerda com a Entropia dos pontos de vista, e à direita, com a Qualidade Kullback-Liebler. Todas as faces do cubo estão divididas em dois polígonos, à excepção de uma, que está dividida em 32 polígonos. A Entropia dos pontos de vista mostra a face com 32 polígonos. No entanto, a Qualidade Kullback-Liebler não a mostra, dando importância apenas a uma visão bem balanceada do cubo.

Melhor ponto de vista de um cubo de acordo com a Entropia dos pontos de vista

Melhor ponto de vista de um cubo de acordo com a Qualidade Kullback-Liebler

Figura 2.22: Melhor vista de uma cubo de acordo com a Entropia dos pontos de vista e de acordo com a Qualidade Kullback-Liebler

Fonte: Mateu Sbert, Dimitri Plemenos, Miquel Feixas, Francisco Gonzalez. Viewpoint Quality: Measures and Applications. Computational Aesthetics in Graphics, Visualization and Imaging, Girona, May 2005, pp.1-8.

Portanto, caso se esteja interessado numa visão bem balanceada da cena ou objectos a observar, a Qualidade Kullback-Liebler será preferível. No entanto, se se valorizar os pormenores, a Entropia dos pontos de vista fornecerá melhores resultados.

2.2.2.3. Qualidade baseada na curvatura das superfícies

Dmitry Sokolov e Dimitri Plemenos propuseram medir a qualidade de um ponto de vista baseando-se na forma como uma face se liga às outras faces que lhe são adjacentes. Quantas mais dobras um ponto de vista conseguir visualizar, na malha de polígonos, melhor será esse ponto de vista, transmitindo-se uma maior quantidade de informação ao utilizador. Para tal, é necessário medir a curvatura das superfícies [Soko05].

É possível medir a curvatura total de uma malha de polígonos através da curvatura extrínseca ou da curvatura intrínseca.

A curvatura extrínseca é medida com base nas arestas formadas por faces adjacentes. Para medir a curvatura extrínseca, Dmitry Sokolov e Dimitri Plemenos sugerem a utilização da Equação (2.10) [Soko05].

$$C_{ext} = \sum_{e \in E} (1 - \cos \Theta_e) \cdot |e| \tag{2.10}$$

E é o conjunto de arestas da cena.

$|e|$ é o comprimento da aresta.

Θ é o ângulo entre as normais às faces que partilham a aresta e.

A curvatura intrínseca é medida com base nos vértices. Para tal, recorre-se à Equação (2.11) [Soko05].

$$C_{\text{int}} = \sum_{v \in V} \left| 2\pi - \sum_{\alpha_i \in \alpha(v)} \alpha_i \right| \tag{2.11}$$

V é o conjunto de vértices da cena.

$\alpha(v)$ é o conjunto de ângulos adjacentes ao vértice v.

Fazendo variar i obtêm-se os vários ângulos, α_i, adjacentes ao vértice v, tal como se pode observar na Figura 2.23.

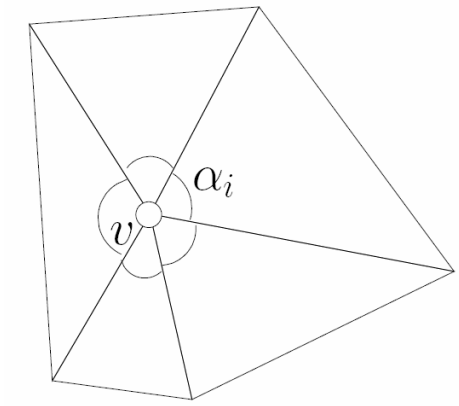


Figura 2.23: Elementos para o cálculo da curvatura intrínseca

Adaptado de: Dmitry Sokolov e Dimitri Plemenos. Viewpoint quality and scene understanding. The 6th International Symposium on Virtual Reality, Archaeology and Curtural Heritage – VAST 2005.

Deste modo, a qualidade um ponto de vista, com base na curvatura das superfícies, pode ser calculada com base na Equação (2.12) [Soko05].

$$I(p) = C(F(p)) \tag{2.12}$$

$I(p)$ é a qualidade do ponto de vista *p* com base na curvatura das superfícies.

$F(p)$ são as faces visíveis do ponto de vista *p*.

$C(F(p))$ é a curvatura da malha de polígonos visível do ponto de vista *p*. Para calcular a curvatura pode utilizar-se a Equação (2.10) ou a Equação (2.11).

A qualidade dos pontos de vista baseada na curvatura é pouco sensível à subdivisão das faces, uma vez que os vértices interiores das faces irão dar origem a ângulos iguais a zero.

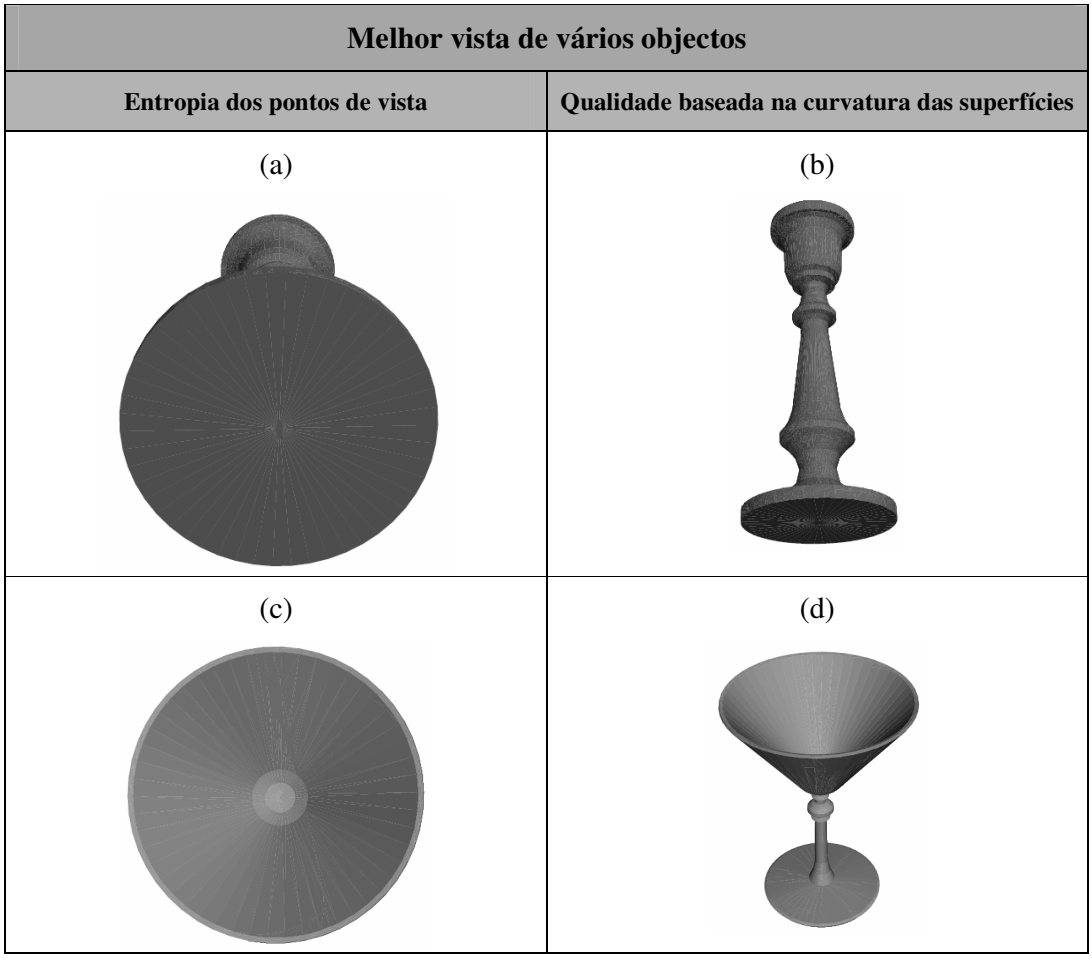


Figura 2.24: Boas vistas de um candelabro e de um copo, de acordo com a entropia dos pontos de vista e de acordo com a qualidade baseada na curvatura das superfícies

Adaptado de: Dmitry Sokolov e Dimitri Plemenos. Viewpoint quality and scene understanding. The 6th International Symposium on Virtual Reality, Archaeology and Curtural Heritage – VAST 2005.

Na Figura 2.24 mostram-se os melhores pontos de vista de dois objectos. Na Figura 2.24 (a) e na Figura 2.24 (c), encontram-se as melhores vistas do candelabro e do copo, de acordo com a entropia dos pontos de vista. A Figura 2.24 (b) e a Figura 2.24 (d) são as melhores vistas do candelabro e do copo, utilizando o critério da qualidade baseada na curvatura das superfícies.

Tanto no caso do candelabro, como no caso do copo, a qualidade medida com base na curvatura das superfícies apresenta melhores resultados, permitindo ao utilizador reconhecer o objecto mais facilmente.

É ainda possível ter, em linha de conta, as áreas projectadas das faces no cálculo da qualidade de um ponto de vista. Para tal, modifica-se a Equação (2.12), com a multiplicação pelo somatório de $P(f)$, sendo $P(f)$ a área projectada da face *f*. A Equação (2.13) é a Equação resultante desta modificação [Soko05].

$$I(p) = C(F(p)) \cdot \sum_{f \in F(p)} P(f) \tag{2.13}$$

Duma forma mais explícita, a Equação (2.13) pode ser apresentada da forma que é indicada na Equação (2.14) [Soko06].

$$I(p) = \sum_{v \in V(p)} \left| 2\pi - \sum_{\phi_i \in \phi(v)} \phi_i \right| \cdot \sum_{f \in F(p)} P(f) \tag{2.14}$$

$F(p)$ são as faces visíveis do ponto de vista *p*.

$P(f)$ é a área projectada da face *f*.

$V(p)$ são os vértices visíveis a partir do ponto de vista *p*.

$\phi(v)$ são os ângulos adjacentes ao vértice *v*.

2.2.2.4. Escolha da melhor forma de avaliar a qualidade de uma vista

Nesta secção escolheremos o critério de avaliação da qualidade de uma vista que melhor se adapta à visita guiada a um museu.

O critério de maximização do número de faces visíveis e o critério de maximização da área projectada podem ser, à partida, eliminados. Tratam-se, em ambos os casos, de critérios simplistas, que nem sempre apresentam ao utilizador uma vista que lhe permite identificar o objecto que está a ser representado.

A minimização das faces degeneradas também é não é boa escolha, porque não contempla o problema das oclusões.

A qualidade Kullback-Leibler escolhe uma vista que proporcione uma visão balanceada da cena. No entanto, este critério corre o risco de deixar escapar pormenores, devido à sua insensibilidade às zonas em que a subdivisão das faces é elevada. Ora, num museu, os pormenores são importantes, pelo que este critério também pode ser posto de parte.

A qualidade baseada na curvatura das superfícies é um critério que proporcionou excelentes resultados, tanto na visualização do candelabro, como na visualização do copo. É visível, na Figura 2.24, que este critério superou a entropia dos pontos de vista para ambos os objectos. Se o museu a visitar fosse um museu com esculturas, escolher-se-ia este critério. No entanto, o museu em que se efectuará a visita é um museu apenas de pinturas. Ora, numa pintura, a curvatura da superfície é muito menos relevante do que numa escultura. Numa pintura, o mais importante são os pormenores. E este critério, à semelhança da qualidade Kullback-Leibler, também é pouco sensível à subdivisão das faces.

Uma vez que a entropia dos pontos de vista é o critério que dá maior importância aos pormenores, este é o critério que melhor se adapta à situação particular da visita guiada a um museu de pintura.

2.2.3. Selecção do melhor ponto de vista de um objecto

Para escolher o melhor ponto de vista de um objecto posiciona-se a câmara numa série de posições correspondentes a pontos na superfície de uma esfera que envolve esse objecto, tal como se mostra na Figura 2.25. A câmara está sempre apontada para o centro da esfera.

Em cada uma das posições, a câmara captura uma vista do objecto. O número de vistas a capturar e a distância da câmara ao objecto são escolhidos pelo utilizador.

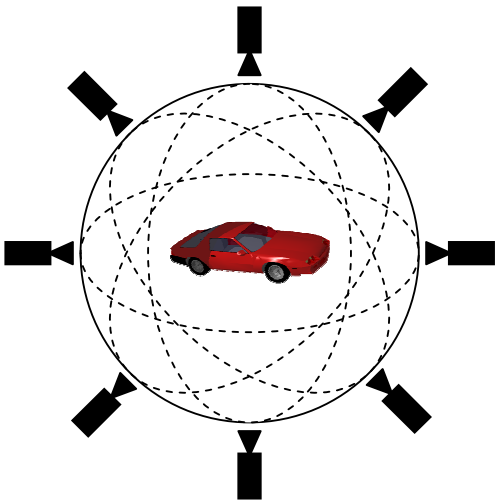


Figura 2.25: Obtenção da melhor vista através do posicionamento da câmara em pontos colocados regularmente numa superfície esférica que envolve o objecto

De entre todas as vistas capturadas pela câmara, a que tem melhor qualidade é escolhida como sendo a que melhor representa o objecto. Para avaliar a qualidade de uma vista pode-se recorrer a uma das estratégias apresentadas na secção 2.2.2.

A melhor vista poderia, por exemplo, ser utilizada numa galeria de modelos de objectos tridimensionais disponíveis na Internet. A melhor vista do objecto ajudaria o utilizador a decidir se queria fazer o *download* do modelo do objecto ou não.

Quanto maior for o número de posições, maior será a probabilidade de se obter, efectivamente, a melhor vista possível do objecto. No entanto, um número mais elevado de posições significa também um tempo de execução mais lento.

Pere Pau Vázquez sugere um método de acelerar a selecção do melhor ponto de vista através da computação adaptativa das melhores vistas. Este método pode ser consultado em [Vázq03a]. Dmitry Sokolov e Dimitri Plemenos apresentam outro método, baseado na visibilidade dos vértices, em [Soko05].

Se o objecto for simples, um único ponto de vista é suficiente. No entanto, para objectos mais complexos, uma vista, ainda que seja a melhor vista, não fornece ao utilizador toda a informação de que ele necessita. No caso da visita guiada, que é objecto desta tese, uma única vista não basta. Para efeitos da visita guiada será necessário escolher, não uma, mas várias boas vistas do modelo. Na próxima secção iremos ver como isto poderá ser feito.

2.2.4. Selecção das melhores pontos de vista de um modelo

Um modelo com várias células, que comunicam entre si através de portais, não pode ser correctamente visualizado através de um único ponto de vista. Para que o utilizador possa apreciar todo o modelo, é necessário escolher vários pontos de vista, e não apenas um. Os pontos de vista escolhidos devem ser os melhores.

Para seleccionar os melhores pontos de vista pode-se colocar a câmara, de forma regular, em várias posições pertencentes à superfície de uma esfera que envolve o objecto, tal como se mostra na Figura 2.25. Depois, de entre essas posições, escolhem-se as que se considerarem melhores. Esta forma de exploração tem o nome de **exploração global**. Na exploração global a câmara fica sempre fora do objecto [Soko06].

No entanto, no caso de modelos de edifícios, a exploração global não é a mais adequada porque não permite que o utilizador se desloque dentro do modelo. Ora, nesta dissertação, está implícito que o modelo a visitar é um edifício, ou parte de um edifício. Por isso, no contexto desta dissertação, a forma de exploração mais adequada é a **exploração local**. Na exploração local a câmara pode ser colocada dentro do modelo a visitar [Soko06]. Uma vez que o nosso modelo está organizado em *voxels*, o centro de cada um desses *voxels* corresponde a um ponto de vista que é um candidato a ser incluído entre os melhores pontos de vista.

Esclarecido qual é o universo de pontos de vista candidatos, há agora que saber como escolher os melhores de entre esse universo. Nesta secção, iremos apresentar e analisar diversas estratégias que permitem decidir quais são os melhores pontos de vista, de entre um conjunto de pontos de vista candidatos.

2.2.4.1. Escolha dos primeiros pontos com qualidade mais elevada

Uma estratégia simples para escolher os melhores pontos de vista de um modelo consistiria em seleccionar aqueles onde a qualidade é mais elevada [Vázq03a]. Por exemplo, podia-se escolher os dez pontos com valores mais elevados de entropia, recorrendo à entropia dos pontos de vista.

Esta estratégia é ineficiente porque teria tendência para escolher pontos apenas na parte mais interessante do modelo, o que levaria a que existissem repetições na informação mostrada ao

utilizador [Vázq03a]. Isto porque é provável que os pontos com qualidade mais elevada estejam próximos uns dos outros. Se um determinado ponto mostra ao utilizador grande quantidade de informação, é provável que o ponto mesmo ao lado dele também mostre muita informação, sendo, dessa forma, também incluído no conjunto dos melhores pontos de vista. No entanto, a diferença entre o que se vê a partir do primeiro ponto e o que se vê a partir do segundo ponto não é grande.

Além disso, ao escolher pontos de vista apenas na parte mais interessante do modelo, existem outras partes do modelo que são completamente ignoradas e que também mereceriam alguma atenção [Vázq03a]. Vamos supor que pretendíamos escolher dez pontos de vista para visualizar um modelo de um edifício já organizado em células e portais. Se os dez melhores pontos estivessem todos dentro da mesma célula, as restantes células nunca chegariam a ser visitadas, o que é negativo.

É, deste modo, importante escolher um conjunto de pontos que tenham qualidade elevada, mas que também mostrem ao utilizador informação diversificada. Seguidamente, iremos apresentar estratégias que permitem concretizar este objectivo.

2.2.4.2. Número de faces visíveis em cada ponto de vista

Esta estratégia escolhe os melhores pontos de vista tendo em conta o número de faces visíveis em cada ponto de vista. Para tal, procede-se da seguinte maneira [Vázq03a, Vázq01a]:

1. Calcular a qualidade da vista para cada um dos pontos de vista do modelo.
2. Registar quais são as faces visíveis em cada um dos pontos. Para tal, pode recorrer-se a um mapa de bits. Serão necessários tantos mapas de bits quanto o número de pontos de vista.
3. Ordenar os pontos de vista por ordem decrescente da qualidade.
4. Criar uma variável *Vistas*, em que são registadas as faces já vistas. Esta variável pode ser também um mapa de bits. De início, ainda nenhuma das faces do modelo foi visualizada.
5. Criar uma variável *Melhores*, que é o conjunto dos melhores pontos de vista. Por agora, ainda nenhum dos pontos de vista faz parte deste conjunto.
6. Atribuir à variável *i* o valor zero.
7. Atribuir à variável *continuar* o valor verdadeiro.
8. Enquanto *i* for menor do que o total de pontos de vista e enquanto *continuar* for igual a verdadeiro, fazer o seguinte:
   - 8.1. Contar quantas das faces visíveis a partir do ponto *i* ainda não estão registadas na variável *Vistas*. Estas são as faces ainda não visualizadas.
   - 8.2. Se o número de faces ainda não visualizadas a partir do ponto *i* for superior ou igual a um *mínimo*, fazer o seguinte:
     - 8.2.1. O ponto *i* passa a fazer parte do conjunto dos *Melhores*.
     - 8.2.2. Adicionar à variável *Vistas* as faces visualizadas a partir do ponto *i*, que ainda não estejam registadas em *Vistas*.
     - 8.2.3. Se o número de faces registadas em *Vistas* for superior ou igual a uma *tolerância* é atribuído o valor falso à variável *continuar*.
   - 8.3. Incrementar *i* em 1 valor.

Através desta estratégia, consegue-se a selecção de pontos de vista com qualidade elevada e através dos quais são visíveis novas faces [Vázq03a]. A variável *mínimo* assegura que apenas os pontos que mostram um número suficiente mínimo de faces novas ao utilizador são admitidos no conjunto dos melhores pontos de vista do modelo.

Vamos supor que, a partir do ponto de vista com melhor qualidade, são visíveis cinco faces do modelo, faces essas que identificaremos aqui por A, B, C, D e E. Se *mínimo* for igual a 3, isso significa que o próximo ponto tem de mostrar, pelo menos, três novas faces.

Sendo assim, se o ponto com a segunda melhor qualidade permitir visualizar as faces A, B, C, D e F, esse ponto não é admitido no conjunto dos melhores pontos de vista, porque apenas mostra 1 face nova, a F.

Agora, vamos supor que a partir do ponto com terceira melhor qualidade se consegue visualizar as faces G, H e I. Este ponto já é admitido no conjunto dos melhores pontos de vista porque mostra 3 novas faces, o que é um valor igual ao estabelecido em *mínimo*.

Quanto maior for a variável *mínimo*, menor será a repetição de faces no conjunto dos melhores pontos de vista, o que é benéfico. Por outro lado, se a variável *mínimo* for elevada, corre-se o risco de não visualizar algumas partes do modelo.

Através da variável *tolerância*, pode-se encurtar o tempo de execução desta estratégia. Logo que o número de faces registadas em *Vistas* seja superior a um determinado limite, não se admitem mais pontos no conjunto dos melhores pontos de vista. Por exemplo, pode estabelecer-se que após se terem visualizado, pelo menos, 90% das faces do modelo, não é necessário acrescentar mais pontos ao conjunto dos melhores pontos de vista.

Esta estratégia proporciona vistas de boa qualidade, tal como se pode observar na Figura 2.26.

Na Figura 2.26 a qualidade dos pontos de vista foi calculada com recurso à Entropia dos pontos de vista. À semelhança do que aconteceu na Figura 2.25, a câmara foi colocada numa série de posições correspondentes a pontos na superfície de uma esfera que contém o objecto. A câmara está sempre apontada para o centro da esfera. Dentre as várias posições da câmara, foram escolhidas as melhores, através da aplicação da estratégia do número de faces visíveis em cada ponto de vista.

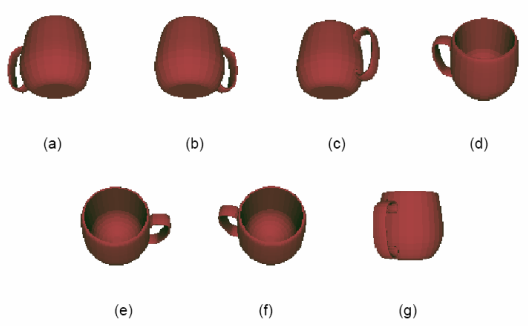


Figura 2.26: Melhores vistas de um objecto, obtidas através da estratégia do número de faces visíveis em cada ponto de vista
Fonte: Pere Pau Vázquez Alcocer. On the selection of good views and its application to computer graphics – Phd Dissertation, Barcelona (Espanha), 2 de Abril de 2003

Existe ainda outra estratégia que recorre à distância entre as áreas das faces de dois pontos de vista. Uma outra estratégia usa a distância entre as entropias de dois pontos de vista. Não abordaremos aqui estes dois métodos, porque existe outro que proporciona melhores resultados e que é baseado em cálculos sucessivos da entropia [Vázq03a, Vázq03b]. Iremos abordá-lo seguidamente.

2.2.4.3. Cálculos sucessivos da entropia

Esta estratégia de selecção dos melhores pontos de vista recorre a cálculos sucessivos da entropia dos pontos de vista, tendo em conta apenas as faces ainda não visualizadas. Para tal, procede-se da seguinte maneira [Vázq03a, Vázq03b]:

1. Criar uma variável *Vistas* em que são registadas as faces já vistas. Por agora, ainda nenhuma das faces do modelo foi visualizada.
2. Criar uma variável *Melhores*, que é o conjunto dos melhores pontos de vista. Por agora, ainda nenhum dos pontos de vista faz parte deste conjunto.
3. Calcular a entropia de todos os pontos de vista não integrados no conjunto dos *Melhores* e tendo em conta apenas as faces que ainda não foram visualizadas, ou seja, que não estão registadas em *Vistas*. As faces já incluídas em *Vistas* passam a ser consideradas como fundo.
4. Ordenar os pontos de vista por ordem decrescente da entropia e seleccionar o ponto com entropia mais elevada para integrar o conjunto dos melhores pontos de vista.
5. Registar, em *Vistas*, as novas faces que são visíveis a partir do ponto seleccionado.
6. Se, através da consulta da lista *Vistas*, se verificar que a percentagem de faces já visualizadas é inferior a uma *percentagem mínima*, então, voltar ao passo 3.

Caso se pretenda que o conjunto dos melhores pontos de vista escolhidos visualize todas as faces, então a *percentagem mínima* deve ser igual a 100%. Se se escolher uma *percentagem mínima* inferior a 100%, esta estratégia tenderá a ter um tempo de execução mais rápido, mas algumas faces poderão não chegar a ser visualizadas.

Uma vez que o cálculo da entropia tem em conta apenas as faces ainda não visualizadas, assegura-se que os pontos escolhidos proporcionam elevada entropia para informação que é nova para o utilizador.

Repare-se que, de acordo com o procedimento enunciado, uma face é considerada como vista desde que a área, dessa face, projectada na superfície da esfera cujo centro é o ponto de vista seleccionado, seja superior a zero. Ora, eventualmente essa área projectada pode ser muito pequena. Ou seja, efectivamente, a face foi vista, mas não foi vista da melhor maneira.

Para refinar esta estratégia, pode-se exigir que, para que uma face seja considerada como vista, a visibilidade dessa face, num ponto de vista, tem de ser superior ou igual a uma percentagem da visibilidade máxima da face. A visibilidade máxima de uma face é igual ao valor mais elevado de visibilidade, dessa face, em todos os pontos de vista [Vázq03a]. Recordamos que a visibilidade de uma face, de acordo com a Equação (2.6) é igual a A_i dividido por A_t, sendo A_i a área projectada da face i na superfície de uma esfera centrada no ponto de vista e sendo A_t a área total dessa esfera.

Sendo assim, podia exigir-se, por exemplo, que, para que uma face fosse considerada como vista, a partir do ponto de vista seleccionado, a visibilidade da face tinha de ser superior ou igual a 60% da visibilidade máxima dessa mesma face.

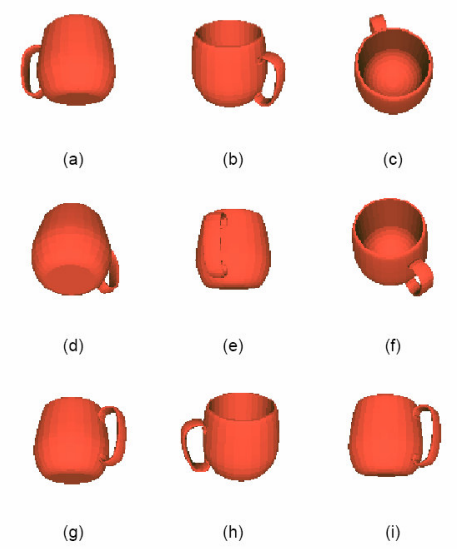


Figura 2.27: Melhores vistas de um objecto, obtidas através da estratégia dos cálculos sucessivos da entropia
Fonte: Pere Pau Vázquez Alcocer. On the selection of good views and its application to computer graphics – Phd Dissertation, Barcelona (Espanha), 2 de Abril de 2003

Na Figura 2.27 mostra-se o resultado desta estratégia para a visualização de um objecto. A câmara foi colocada numa série de posições correspondentes a pontos na superfície de uma esfera que envolve o objecto. A câmara está sempre apontada para o centro da esfera. Dentre as várias posições da câmara, foram escolhidas as melhores, através da aplicação da estratégia dos cálculos sucessivos da entropia.

Existe ainda mais uma estratégia de selecção dos melhores pontos de vista de um modelo baseada na Distância Kullback-Leibler entre dois pontos de vista [Sber05]. Não abordaremos aqui esta estratégia porque, por uma questão de coerência, convém utilizá-la em conjunto com a Qualidade Kullback-Leibler e já se estabeleceu que a medida de qualidade de uma vista que se iria utilizar é a entropia dos pontos de vista. Tanto a Distância Kullback-Leibler como a Qualidade Kullback-Leibler são da autoria de Mateu Sbert, Dimitri Plemenos, Miquel Feixas e Francisco González.

2.2.4.4. Escolha e adaptação da estratégia de selecção dos melhores pontos de vista

Nesta secção escolheremos a estratégia de selecção dos melhores pontos de vista mais vantajosa para a visita guiada ao museu.

A estratégia de escolha dos primeiros pontos com qualidade mais elevada pode, à partida, ser posta de parte devido à sua tendência para mostrar informação repetida ao utilizador. Além disso, esta estratégia pode levar a que partes interessantes do modelo nunca cheguem a ser visualizadas.

A estratégia do número de faces visíveis em cada ponto de vista exige que a entropia seja calculada apenas uma vez para cada um dos pontos de vista candidatos.

Na estratégia de cálculos sucessivos da entropia, a entropia para cada ponto de vista candidato pode ter de ser calculada mais do que uma vez. Isto a não ser que o primeiro ponto de vista seleccionado permita visualizar correctamente todas as faces, o que será uma situação rara.

Portanto, a estratégia de cálculos sucessivos da entropia é mais exigente, no que se refere à capacidade de processamento do computador.

A estratégia de cálculos sucessivos da entropia permite estabelecer que, para que uma face seja considerada como visualizada, a visibilidade dessa face tem de ser superior ou igual a uma percentagem da visibilidade máxima dessa mesma face. Este aspecto é útil, caso se pretenda ser mais exigente na correcta visualização do objecto ou cena.

Uma vez que a estratégia de cálculos sucessivos da entropia é mais versátil, é esta a estratégia escolhida.

Como os modelos geométricos, que são objecto desta tese, estão organizados em células, são necessárias ligeiras adaptações à estratégia dos cálculos sucessivos da entropia. Aliás, esta organização em células facilita a escolha dos melhores pontos de vista, porque permite que se trabalhe com o modelo célula e célula, e não como um todo. Sendo assim, o que se irá fazer é seleccionar os melhores pontos de vista para cada uma das células constituintes do modelo. Desta forma, para cada uma das células, repete-se o seguinte procedimento [Andú04]:

1. Calcular a entropia dos pontos de vista para cada um dos *voxels* da célula. A entropia é calculada no ponto correspondente ao centro do *voxel*. Guardar também, para cada um desses pontos, a área projectada visível de cada face do modelo.

   Caso a câmara esteja sempre à mesma altura, bastará calcular a entropia para os *voxels* da fatia horizontal que está à altura da câmara. Para efeitos do cálculo da entropia, é possível saber qual é a área projectada visível das faces atribuindo, a cada uma delas, uma cor diferente.

   Após se ter calculado a entropia e a área projectada visível das faces para cada um dos *voxels* da célula, fica-se a saber quais são as faces visíveis a partir dessa célula. Uma face do modelo é visível a partir de uma célula se, pelo menos em um dos *voxels* dessa célula, a área projectada dessa face for maior do que zero.

2. Seleccionar o *voxel* com entropia mais elevada, incluí-lo no conjunto dos melhores pontos de vista e registar quais são as faces que são visíveis a partir do centro desse *voxel*. O valor de entropia do *voxel* seleccionado mede a **relevância da célula**. Quanto maior o valor máximo de entropia encontrado, mais relevante é a célula. Uma célula com elevada relevância tem muita informação para mostrar ao utilizador.

3. Se todas as faces visíveis puderem ser observadas a partir do *voxel* com melhor entropia, então esse *voxel* é suficiente para permitir uma correcta visualização da célula. Senão, será necessário voltar a calcular a entropia para cada um dos *voxels* da célula, com excepção do *voxel* com melhor entropia. Esta segunda entropia será calculada tendo em conta apenas as faces visíveis que não foram observadas no *voxel* de melhor entropia. As faces já vistas, para efeitos do cálculo da entropia, passam a ser consideradas como fundo. Depois, volta-se a escolher o *voxel* com melhor entropia, que ainda não foi incluído no conjunto dos melhores pontos de vista. Se o primeiro *voxel* escolhido e o segundo *voxel*

escolhido forem capazes de visualizar todas as faces visíveis da célula, então esses dois *voxels* serão suficientes. Senão, será necessário voltar a calcular a entropia para os *voxels* o número de vezes suficiente até que todas as faces da célula tenham sido visualizadas.

Pode-se exigir que o conjunto final de *voxels* escolhidos tem de permitir observar todas as faces visíveis. No entanto, caso se ache que não é grave que algumas das faces sejam excluídas da observação, pode-se estabelecer uma percentagem mínima de faces a observar.

Após se ter efectuado este procedimento para cada uma das células do modelo, obter-se-á, para cada uma delas, o seu valor de relevância e um ou mais *voxel*s a partir dos quais são observáveis as faces visíveis da célula. Os centros destes *voxels* são os **melhores pontos de vista** das células.

À semelhança do que acontece na estratégia dos cálculos sucessivos da entropia, pode-se considerar que uma face só é considerada como vista caso a sua área projectada seja superior ou igual a uma percentagem da visibilidade máxima dessa mesma face.

Caso todas as faces da célula tenham igual importância, o processo apresentado produz bons resultados.

No entanto, na realidade que nos rodeia, nem todos os objectos têm a mesma importância. Através da nossa experiência de vida, aprendemos a que porções da realidade devemos prestar mais atenção. Sendo assim, sabemos, ao visitar um museu, que as obras expostas são merecedoras de uma observação demorada. No entanto, uma simples cadeira, onde os visitantes podem descansar alguns minutos durante a visita, não é objecto que mereça atenção especial. Infelizmente, não existe uma forma de ensinar um computador a saber reconhecer algo de tão abstracto como uma obra de arte.

A estratégia de cálculos sucessivos da entropia tenderá a seleccionar *voxels* onde se visualiza um elevado número de faces, independentemente da qualidade dessas faces. Isso leva a que uma face de uma cadeira tenha exactamente a mesma importância que a face de uma pintura de Leonardo da Vinci.

Se se quiser que a pintura de Leonardo da Vinci receba maior atenção, pode-se atribuir um valor de importância mais elevado às faces que pertencem a esta pintura. No caso da visita guiada a um museu, atribuir-se-ia um valor de importância mais elevado às obras que estão em exposição [Andú04].

Poderá decidir-se visitar apenas as células que contenham faces com elevado valor de importância. Voltemos ao exemplo da visita guiada ao Museu Nacional de Arte Antiga. Se o utilizador estivesse interessado apenas em pintura portuguesa do século XV, as faces de objectos inseridos dentro desta categoria receberiam um valor de importância mais elevado. Dessa forma, a visita guiada passaria obrigatoriamente pelas células onde se encontrasse exposta pintura portuguesa do século XV, não perdendo tempo em visitar as restantes células.

Poderá também optar-se por visitar só as células com elevada relevância. Desse modo, para que a célula seja visitada, o seu valor de relevância deve estar acima de um determinado limite. Ou seja, a célula tem de transmitir, pelo menos, uma certa quantidade de informação ao utilizador para que mereça a pena visitá-la [Andú04].

2.3. Criação do percurso

Como já referimos anteriormente, a visita guiada através de um modelo começa pela organização desse modelo em *voxels*. Depois, a partir desse *voxels* podemos dividir o modelo em células. A comunicação entre as células faz-se através dos portais. A partir daqui é possível construir um grafo de células e portais que representa o modelo. Depois, avalia-se a relevância de cada célula e escolhe-se para cada uma delas os melhores pontos de vista. As células a incluir na visita são as que tiverem uma relevância superior a um determinado mínimo ou que contiverem faces com elevado valor de importância. A seguir, há que criar um percurso que passe por todos os pontos de vista seleccionados nas células que se decidiu incluir na visita. É isso que iremos abordar nesta secção.

Para a criação do percurso é necessária a geração do caminho de alto nível e do caminho de baixo nível [Andú04].

- **Caminho de alto nível**
  - O caminho de alto nível é um conjunto de células dispostas pela ordem através da qual o utilizador às irá percorrer. Na Figura 2.28 encontra-se um corte horizontal de um modelo. O percurso deve passar, obrigatoriamente, pelas células 0, 4 e 5, que são as células relevantes. A ordem de visita poderia ser: 4, 2, 0 e 5. Este seria o percurso de alto nível.

    Embora não pertença ao grupo das células a incluir na visita, a célula 2 também foi incluída, uma vez que é necessário atravessá-la, de forma a conseguir chegar da célula 4 à célula 0.

- **Caminho de baixo nível**
  - Para cada uma das células visitadas é construído um caminho de baixo nível. O caminho de baixo nível deverá passar por todos os melhores pontos de vista dessa célula. Os melhores pontos de vista são os pontos que proporcionam boas vistas, permitindo uma boa observação da célula. Após isso, o caminho de baixo nível deverá levar o utilizador até um portal que permita o acesso à próxima célula a visitar.

    Na Figura 2.28 está assinalado, com linhas a traço interrompido e linhas a ponteado, o caminho de baixo nível dentro de cada célula.

    Eventualmente, existirão, no caminho de alto nível, células que foram incluídas apenas para permitirem a passagem entre as diferentes células. É o caso da célula 2, da Figura 2.28. Não é necessário passar pelos melhores pontos de vista deste tipo de células.

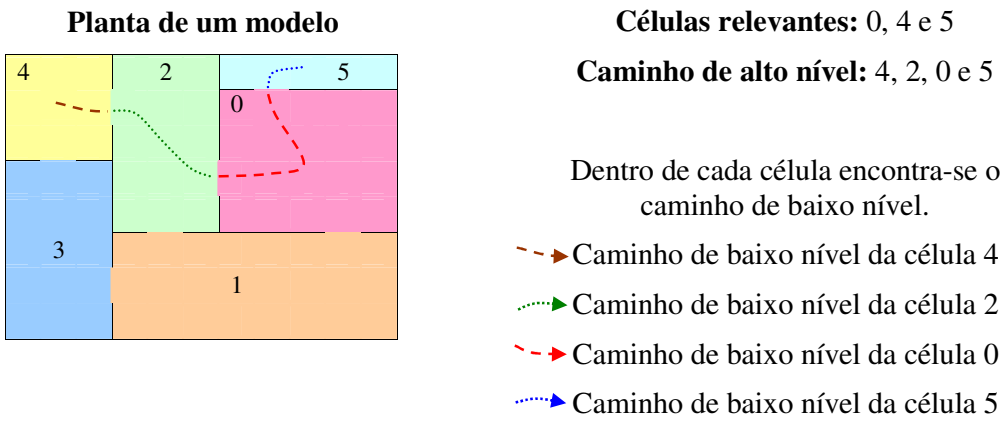


Figura 2.28: Exemplo de um caminho de alto nível e de um caminho de baixo nível para um modelo

As células do caminho de alto nível devem ser escolhidas e ordenadas de maneira a que o percurso seja o mais curto possível. Por exemplo, no caso da Figura 2.28, o caminho de alto nível poderia ser: 4, 3, 1, 0 e 5. Este caminho passa por todas as células relevantes, sendo desse modo um caminho válido. No entanto, este caminho inclui duas células, a 1 e a 3, que não são relevantes para o utilizador e que apenas o fazem perder tempo, não lhe mostrando nada de interessante. O caminho de alto nível 4, 2, 0 e 5 inclui apenas uma única célula não relevante (célula essa à qual é impossível escapar, por ser um ponto de passagem entre a célula 4 e a célula 0, tal como já foi anteriormente referido), reduzindo dessa forma o tempo que o utilizador perde com as partes desinteressantes do modelo. Por isso, quanto mais curto for o caminho de alto nível, menos células não relevantes são percorridos e, consequentemente, mais interessante é o percurso para o utilizador.

No caminho de baixo nível segue-se uma lógica semelhante. Há que encontrar uma forma de fazer chegar o utilizador às posições que correspondem aos melhores pontos de vista da forma mais rápida possível, evitando assim perder tempo com posições que pouco ou nada têm para mostrar. Para isso, é necessário encontrar o caminho mais curto até aos melhores pontos de vista da célula.

Para saber qual é o caminho de alto nível mais curto e o caminho de baixo nível mais curto, é necessário recorrer aos algoritmos de pesquisa, uma área da Inteligência Artificial. Na próxima secção iremos abordar este tipo de algoritmos.

2.3.1. Algoritmos de pesquisa

Os algoritmos de pesquisa pertencem à área da Inteligência Artificial. De acordo com Stuart Russel, estes algoritmos permitem resolver um problema que é definido por quatro componentes [Russ03]:

- Estado inicial
  - O estado inicial é o primeiro estado do problema.

- Função sucessor
  - A função sucessor indica quais as acções que estão disponíveis num determinado estado. A função sucessor retorna um conjunto de pares ordenados. Cada par contém uma **acção** e o estado que se atinge após se executar essa acção.
  - A função sucessor e o estado inicial definem o **espaço de estados**. O espaço de estados é o conjunto de todos os estados a que se pode chegar a partir do estado inicial. O espaço de estados pode ser representado através de um grafo em que os vértices são os estados e os ramos entre os vértices são as acções.
  - Um **caminho**, no espaço de estados, é uma sequência de estados conectada por uma sequência de acções.

- Teste do objectivo
  - O teste do objectivo permite determinar se um estado é o estado objectivo. Ao atingir o estado objectivo, encontra-se uma solução para o problema.

- Custo do caminho
  - A cada caminho é atribuído um custo. O custo do caminho é um valor numérico.
  - Cada acção, que leva o utilizador de um estado a outro, tem o chamado **custo do passo**. O custo do passo é também um valor numérico. Para saber qual o custo de um caminho somam-se os custos associados a cada uma das acções desse caminho.

É a partir destes quatro componentes que o algoritmo de pesquisa vai conseguir retornar uma **solução**. Uma solução para um problema é um caminho que começa no estado inicial e que, através de uma sequência de acções, permite chegar ao estado objectivo.

As componentes de um problema podem ser mais fáceis de compreender através de um exemplo. Tomemos o caso da Figura 2.29 em que se apresenta o corte horizontal de um modelo ao qual foi sobreposto o grafo de célula e portais. Para cada ramo é indicada a distância, em metros, que se percorre deste o centro de uma célula até ao centro de outra célula. Para o cálculo desta distância são tomados em linha de conta os portais entre as células, tal como é possível observar no ramo entre a célula 0 e a célula 5. Neste exemplo em particular, considerou-se que, ao movimentar-se da célula 0 para a célula 5, o utilizador se desloca 10 metros para baixo e, depois, 20 metros para a direita, o que totaliza 30 metros. Um modo de proceder semelhante foi adoptado para o ramo que une a célula 4 à célula 5. Suponhamos que o utilizador se encontra na célula 0 e pretende chegar à célula 4. As componentes para este problema são [Russ03]:

- Estado inicial
  - Na célula 0.

- Função sucessor
  - A função sucessor aplicada ao estado "Na célula 0" retorna os seguintes pares:
    - (Acção: "Ir para a Célula 1", Estado: "Na Célula 1").
    - (Acção: "Ir para a Célula 5", Estado: "Na Célula 5").
  - O espaço de estados é o grafo, sobreposto ao modelo da Figura 2.29.

- Teste do objectivo
  - Atinge-se o objectivo quando o estado for igual a "Na célula 4".

- Custo do caminho
  - O custo do passo das várias acções está registado nos ramos que ligam os vértices do grafo. Por exemplo, a acção que, aplicada ao estado "Na Célula 0", permite atingir o estado "Na Célula 5", tem custo 30. Para saber o custo do caminho efectua-se o somatório do custo do passo associado a cada uma das acções desse caminho.

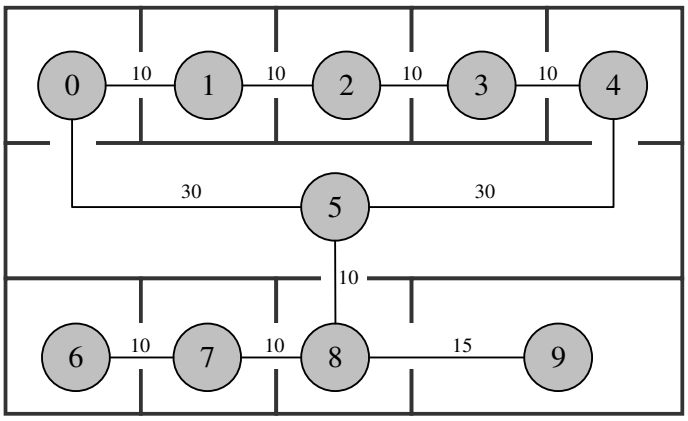


Figura 2.29: Corte horizontal de um modelo com o grafo de células e portais sobreposto

Para se resolver o problema é necessário pesquisar o espaço de estados. Para tal, recorre-se a uma **árvore de pesquisa**. A raiz da árvore de pesquisa corresponde ao estado inicial. Através da aplicação da função sucessor a árvore de pesquisa é sucessivamente expandida. Na Figura 2.30 encontra-se um exemplo de expansão da árvore de pesquisa para o grafo da Figura 2.29. Pretende-se encontrar um caminho da Célula 0 para a Célula 4.

Inicialmente, a árvore de pesquisa contém apenas um nó em que está registado o estado inicial "Na Célula 0". Uma vez que o estado "Na Célula 0" não é o estado objectivo, aplica-se a função sucessor ao estado "Na Célula 0", gerando-se dessa forma dois novos estados: "Na Célula 1" e "Na Célula 5". Seguidamente, tem de se escolher um destes dois estados. Suponhamos que se escolhia o estado "Na Célula 1". Uma vez que se escolheu o estado "Na Célula 1", há que verificar se este é o estado objectivo. Como não é o estado objectivo, aplica-

se novamente a função sucessor ao estado "Na Célula 1". A partir daí, obtém-se dois novos sucessores. Um deles é "Na Célula 2". O outro é "Na Célula 0", porque o portal entre a Célula 0 e a Célula 1 permite que o utilizador se desloque em ambos os sentidos. Seguidamente, escolhe-se um de entre estes dois estados, ou caso se julgue mais proveitoso, pode-se voltar ao estado "Na Célula 5" e efectuar uma nova expansão a partir daqui.

A escolha do estado a expandir é determinada pela **estratégia de pesquisa**. Posteriormente, iremos apresentar várias estratégias de pesquisa. Cada uma delas adopta diferentes formas de escolher qual o estado a expandir.

O processo de escolha, teste e expansão de estado continua até que se atinja o estado objectivo, ou até que não restem mais estados para expandir.

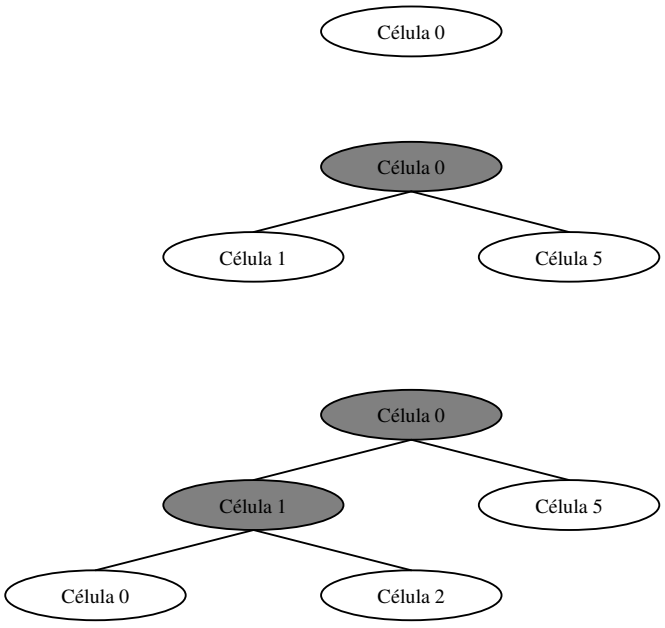


Figura 2.30: Expansão de uma árvore de pesquisa

Há que realçar que a árvore de pesquisa e o espaço de estados são entidades diferentes e distintas. No grafo da Figura 2.29, que representa o espaço de estados, existem 10 estados. No entanto, uma vez que são possíveis, neste grafo, infinitos caminhos, a árvore de pesquisa tem, consequentemente, um número infinito de nós. Por exemplo, o utilizador poderia deslocar-se da Célula 0, para a Célula 1, para a Célula 0, e novamente para a Célula 1, entrando num ciclo que nunca mais terminaria.

Todos os **nós** de uma árvore de pesquisa, à excepção da raiz, são uma estrutura de dados com pai, estado, acção, custo do caminho e profundidade. A raiz tem também estado, acção, custo do caminho e profundidade, mas não tem pai.

Na estrutura de dados do nó, o estado é o estado ao qual o nó corresponde. O pai é o nó que gerou o nó em causa. A acção é aplicada ao estado do pai, para atingir o estado do filho. O custo do caminho refere-se ao custo do caminho que começa no estado inicial e que termina no estado do nó. A profundidade é o número de vezes que a árvore foi expandida até se chegar a este nó.

Os nós da árvore de pesquisa que foram gerados, mas que ainda não foram expandidos, são denominados por **margem**. Cada um dos nós da margem é uma **folha** [Russ03]. Na Figura 2.30, os nós da margem são os que têm fundo branco. De entre os nós da margem, a estratégia de pesquisa escolhe qual o próximo a expandir.

Uma pesquisa numa árvore pode ser executada através da **função de pesquisa genérica numa árvore**. Esta função desenvolve-se ao longo dos seguintes passos [Russ03]:

1. Receber o problema a resolver.
2. Criar um novo nó, correspondente ao estado inicial do problema, e inseri-lo na estrutura de dados *Margem*.
3. Se a *Margem* estiver vazia retornar Falha.
4. Remover o primeiro nó da *Margem.* A este novo nó vamos chamar *Pai*.
5. Efectuar o teste do objectivo no *Pai*. Deste modo, verifica-se se este nó corresponde ao estado objectivo.
6. Se o *Pai* corresponder ao estado objectivo:

   6.1. Retornar a solução do problema proporcionada pelo *Pai*.
7. Aplicar a função sucessor ao *Pai*. Para cada um dos pares gerados pela função sucessor, efectuar o seguinte:

   7.1. Criar um novo nó, a que chamaremos o *Filho*.

   7.2. Preencher a estrutura de dados do *Filho*.

      7.2.1. O estado do *Filho* é igual ao estado contido no par gerado pela função sucessor.

      7.2.2. O pai do *Filho* é, justamente, o nó *Pai*, removido da *Margem* no Passo 4.

      7.2.3. A acção do *Filho* é a acção contida no par gerado pela função sucessor.

      7.2.4. O custo do caminho do *Filho* é igual ao custo do caminho do *Pai* somado ao custo do passo necessário para transitar do estado do *Pai* para o estado do *Filho*.

      7.2.5. A profundidade do *Filho* é igual à profundidade do *Pai* mais 1.

      7.2.6. Adicionar o *Filho* à *Margem*.
8. Voltar ao Passo 3.

De acordo com Russel, uma estratégia de pesquisa é avaliada de acordo com os seguintes parâmetros [Russ03]:

- Completude
  - Uma estratégia de pesquisa é completa se for capaz de encontrar uma solução para o problema. Isto caso a solução exista. Se o problema não tiver solução, a estratégia de pesquisa, por melhor que seja, não será capaz de encontrar aquilo que não existe.
- Optimalidade
  - A estratégia de pesquisa é óptima se conseguir encontrar a solução óptima. A solução óptima é aquela que tem menor custo do caminho.

- Complexidade temporal
  - A complexidade temporal refere-se ao tempo que a estratégia de pesquisa demora até encontrar a solução.
  - Aqui, a complexidade temporal é medida através do número de nós gerados durante a pesquisa.
- Complexidade espacial
  - A complexidade espacial refere-se à quantidade de memória que é necessária para a execução da estratégia de pesquisa.
  - Nesta dissertação, o espaço é medido pelo número máximo de nós armazenados na memória.

A complexidade temporal e a complexidade espacial são determinadas pelas seguintes quantidades [Russ03]:

- Factor de ramificação (r)
  - É o número máximo de sucessores que um nó pode ter na árvore de pesquisa.
- Profundidade da solução mais superficial (p)
  - É a profundidade à qual se encontra o nó mais superficial correspondente ao estado objectivo.
- Profundidade máxima do espaço de estados (m)
  - Corresponde ao comprimento do caminho mais longo que possa existir no espaço de estados. Eventualmente, o caminho mais longo pode ter um comprimento infinito.

Existem dois tipos de estratégias de pesquisa:

- Estratégias de pesquisa ignorantes.
- Estratégias de pesquisa informadas.

As estratégias de pesquisa ignorantes, também denominadas por estratégias de pesquisa cegas, não dispõem de mais informação para além daquela que é fornecida na definição do problema. As estratégias de pesquisa informadas conseguem prever se um estado é mais prometedor do que outro, no se refere à possibilidade de chegar à solução óptima. As estratégias de pesquisa diferenciam-se umas das outras através da ordem pela qual escolhem os nós a expandir.

Nas secções seguintes iremos abordar as estratégias de pesquisa ignorantes e as estratégias de pesquisa informadas.

2.3.1.1. Estratégias de pesquisa ignorantes

As estratégias de pesquisa ignorantes não possuem informação adicional para além daquela que é proporcionada na definição do problema.

Uma das estratégias de pesquisa ignorantes é a **pesquisa em largura primeiro**. Na pesquisa em largura primeiro começa-se por expandir a raiz da árvore de pesquisa. Depois, todos os sucessores da raiz são expandidos. Seguidamente, expande-se todos os sucessores dos sucessores da raiz, e por aí adiante [Pear84, Russ03].

Para implementar a pesquisa em largura primeiro recorre-se à função de pesquisa genérica numa árvore. A estrutura de dados *Margem* desta função é uma fila de espera em que o primeiro nó a entrar é o primeiro nó a sair (FIFO – *First In First Out*). Os sucessores são colocados no fim da fila [Pear84, Russ03]. Na Figura 2.31 mostra-se a evolução da procura numa árvore em que cada nó gera sempre dois sucessores. O nó a ser expandido em cada etapa é destacado com uma linha a traço interrompido.

Caso p (profundidade da solução mais superficial) e r (factor de ramificação) sejam finitos, a pesquisa em largura primeiro é completa, acabando sempre por encontrar uma solução para o problema. A solução encontrada é a solução óptima se o custo do passo for constante. Se o custo do passo não for constante, não é garantido que o primeiro nó objectivo encontrado seja aquele que proporciona a solução com o custo do caminho mais baixo [Pear84, Rich91, Russ03].

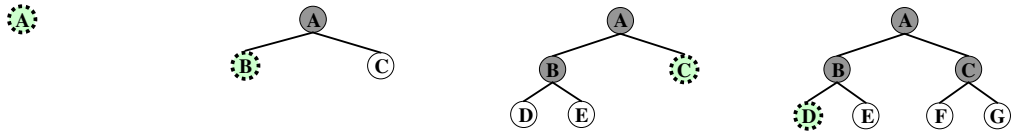


Figura 2.31: Evolução da pesquisa em largura primeiro

Adaptado de: Stuart Russell e Peter Norvig. Artificial Intelligence – A Modern Approach, Segunda Edição. Prentice Hall, Estados Unidos da América, Upper Saddle River, New Jersey, 2003.

Para calcular a complexidade temporal da pesquisa em largura primeiro temos de saber quantos nós são gerados. Se cada nó gerar r sucessores, a raiz da árvore dá origem, no nível 1, a r novos nós. Depois, os nós do nível 1 geram mais r^2 nós, pertencentes ao nível 2. Os nós do nível 2, por sua vez, geram mais r^3 nós para o nível 3, e por aí adiante. No pior caso possível, se a solução mais superficial se encontrar à profundidade p, e supondo que é necessário expandir todos os nós da profundidade p menos o último, que é o nó objectivo, então o total de nós gerados é $r + r^2 + r^3 + \ldots + r^p + (r^{p+1} - r)$. A complexidade temporal da pesquisa em largura primeiro é, dessa forma, $O\left(r^{p+1}\right)$ [Russ03].

Para a complexidade espacial é necessário calcular quantos nós são armazenados na memória. O número de nós armazenados em memória é igual ao número de nós gerados mais 1, que é o nó da raiz. Sendo assim, a complexidade espacial é também $O\left(r^{p+1}\right)$.

Outra estratégia de pesquisa é a **pesquisa de custo uniforme**. A pesquisa de custo uniforme expande em primeiro lugar o nó da margem que tiver custo do caminho mais baixo. Caso os

custos do passo sejam iguais, então a pesquisa de custo uniforme funciona da mesma maneira que a pesquisa em largura primeiro [Pear84, Russ03].

Para que a pesquisa de custo uniforme funcione correctamente, o custo do passo deverá ser sempre superior ou igual a um determinado valor mínimo e não nulo, o valor *min*. Caso contrário, a pesquisa de custo uniforme ficará presa, num ciclo infinito, ao expandir continuamente um nó com uma acção de custo zero que leve continuamente ao mesmo estado. Se o custo do passo for superior a *min*, então garante-se que a pesquisa de custo uniforme é completa e óptima.

Se a solução óptima tiver um custo do caminho c, e se os custos do passo forem sempre superiores ou iguais a *min*, a complexidade temporal e a complexidade espacial serão, no pior caso, $O\left(r^{\lceil c/\min \rceil}\right)$. Este valor pode facilmente tornar-se mais alto do que $O\left(r^{p+1}\right)$, correspondente à complexidade temporal e espacial da procura em largura primeiro [Russ03].

A **pesquisa em profundidade primeiro** é outra estratégia de pesquisa ignorante. Esta estratégia expande em primeiro lugar o nó mais profundo na margem da árvore de pesquisa [Pear84, Russ03]. A evolução deste algoritmo pode ser observada na Figura 2.32 para uma árvore de pesquisa em que o máximo de níveis possíveis é três.

A pesquisa em profundidade primeiro pode ser implementada recorrendo à função de pesquisa genérica numa árvore, com recurso a uma pilha em que o último elemento a entrar seja o primeiro elemento a sair (LIFO – *Last In First Out*). Os sucessores são colocados no topo da pilha [Pear84, Russ03]. Na Figura 2.32, os nós assinalados a negro são os nós descartados da pilha, libertando espaço em memória.

Com uma profundidade máxima do espaço de estados m, e um factor de ramificação r, o número máximo de nós a armazenar em memória é $rm+1$. A complexidade espacial da pesquisa em profundidade primeiro é, desta forma, $O(rm)$.

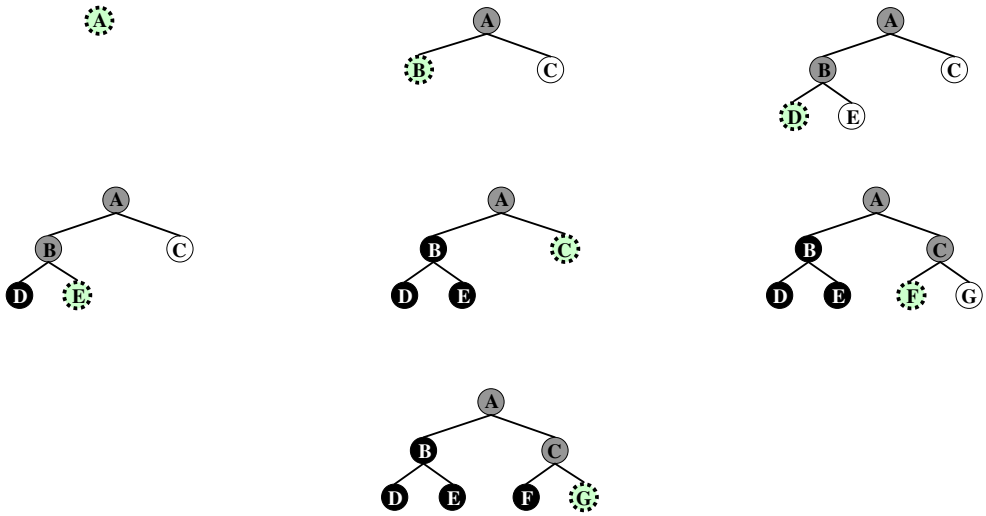


Figura 2.32: Evolução da pesquisa em profundidade primeiro

Adaptado de: Stuart Russell e Peter Norvig. Artificial Intelligence – A Modern Approach, Segunda Edição. Prentice Hall, Estados Unidos da América, Upper Saddle River, New Jersey, 2003.

É possível diminuir as exigências a nível da memória através da **pesquisa com retrocesso**, uma variante da pesquisa em profundidade. Na pesquisa com retrocesso só se gera um sucessor de cada vez. Para que isto possa acontecer é necessário que cada nó parcialmente expandido saiba qual é o próximo nó a gerar. Através da pesquisa com retrocesso é possível reduzir a complexidade espacial para $O(m)$ [Russ03].

A pesquisa em profundidade não é completa, pois caso m seja infinito, uma escolha menos afortunada do nó a expandir pode fazer com que se percorra para sempre um caminho que nunca levará a nenhuma solução. No entanto, se m for finito, então a pesquisa em profundidade é garantidamente completa [Pear84, Rich91, Russ03].

A pesquisa em profundidade também não é óptima. Para verificar esta afirmação, vamos supor que a solução óptima podia ser encontrada no nó C da Figura 2.32. O algoritmo, como se pode ver, explora primeiro toda a parte esquerda da árvore. Caso o nó D providenciasse uma solução não óptima, esta seria a primeira solução a ser encontrada, e portanto seria a solução retornada pelo algoritmo.

No pior caso possível, a pesquisa em profundidade gerará todos os nós da árvore de pesquisa até que a solução seja encontrada, pelo que a complexidade temporal é $O(r^m)$, em que m é a profundidade máxima do espaço de estados. Em comparação com a pesquisa em largura primeiro, a pesquisa em profundidade primeiro pode ser muito mais exigente em termos de tempo se m for maior do que p. Aliás, como já referimos anteriormente, m até pode ser infinito.

Portanto, quando m é infinito a pesquisa em profundidade não é uma boa opção. No entanto, existe outro algoritmo que resolve o problema da profundidade, a **pesquisa de profundidade limitada** [Russ03]. Na pesquisa de profundidade limitada é definido um limite l. Os nós que estão a profundidade l são tratados como se fossem nós sem sucessores. Desta forma, ainda que m seja infinito, graças ao limite l, garante-se que não se percorrerá para sempre um caminho inútil.

A pesquisa em profundidade limitada não é garantidamente completa. Se l for menor que p, não será retornada nenhuma solução. À semelhança da pesquisa em profundidade primeiro, também não há garantias de que a pesquisa de profundidade limitada consiga encontrar a solução óptima. Uma vez que no, pior caso, são armazenados em memória $rl+1$ nós, a complexidade espacial é $O(rl)$. Também, no pior caso possível, é necessário gerar todos os nós até que se atinja o limite, l. Por isso, a complexidade temporal é $O(r^l)$ [Russ03].

Outra estratégia de pesquisa ignorante é a **pesquisa por aprofundamento progressivo**. Neste tipo de pesquisa o limite, l, começa em 0, depois passa para 1, depois para 2, depois para 3 e por aí adiante, até se chegar a p, que é a profundidade da solução mais superficial. Em cada iteração, os nós de profundidade l são tratados como se fossem nós sem sucessores. A evolução da pesquisa por aprofundamento progressivo pode ser observada na Figura 2.33. Se o custo do passo for constante, este algoritmo consegue encontrar a solução óptima [Russ03].

A pesquisa por aprofundamento progressivo acumula as vantagens da pesquisa em largura primeiro e da pesquisa em profundidade primeiro. À semelhança da pesquisa em largura primeiro, a pesquisa por aprofundamento progressivo é completa, e também óptima, caso o custo do passo seja constante. Tal como a pesquisa em profundidade primeiro, este é também um algoritmo pouco exigente em termos de memória, sendo a complexidade espacial de $O(rp)$ [Russ03].

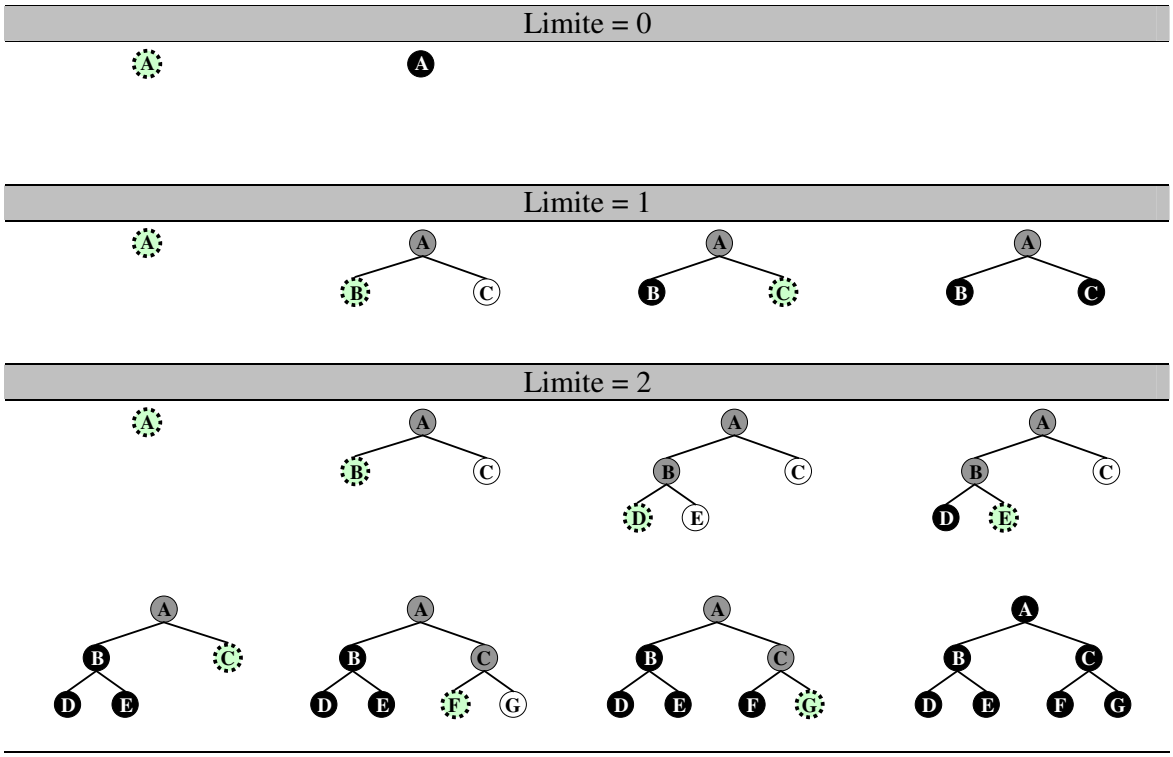


Figura 2.33: Evolução da pesquisa por aprofundamento progressivo

Adaptado de: Stuart Russell e Peter Norvig. Artificial Intelligence – A Modern Approach, Segunda Edição. Prentice Hall, Estados Unidos da América, Upper Saddle River, New Jersey, 2003.

Repare-se que, na pesquisa por aprofundamento progressivo, existem nós da árvore que são gerados várias vezes. Os nós de profundidade p são gerados 1 vez. Os nós de profundidade $p-1$ são gerados duas vezes. Os nós de profundidade $p-2$ são gerados duas vezes, e por aí adiante, até se chegar aos sucessores da raiz, que são gerados p vezes. O número total de nós gerados é, desta forma: $(p)r+(p-1)r^2+\ldots+(1)r^p$, o que resulta numa complexidade temporal de $O(r^p)$ [Russ03].

Lembramos que na pesquisa em largura primeiro o número de nós gerados é: $r+r^2+\ldots+r^p+(r^{p+1}-r)$. Portanto, se r for igual a 10 e p for igual a 5, na pesquisa em largura primeiro geram-se 1111100 nós e na pesquisa por aprofundamento progressivo geram-se 123450 nós, tal como se pode verificar no Quadro 2.2. Deste modo, verifica-se que, apesar da repetição da geração dos nós, a pesquisa por aprofundamento progressivo é mais rápida do que a pesquisa em largura primeiro. Esta maior rapidez deve-se ao facto de a pesquisa em largura primeiro ainda gerar alguns nós à profundidade $p+1$. A pesquisa por aprofundamento progressivo não gera nós à profundidade $p+1$, resultando daí uma economia de tempo [Russ03].

Segundo Stuart Russel, a pesquisa por aprofundamento progressivo é uma boa escolha quando existe um espaço de estados vasto e quando a profundidade da solução mais superficial não é conhecida.

Número total de nós gerados em dois algoritmos de pesquisa	
Algoritmo	**Nós gerados**
Pesquisa por aprofundamento progressivo	$(p)r + (p-1)r^2 + \ldots + (1)r^p$ **Exemplo:** $r = 10$ $p = 5$ Nós gerados $= 5\times10 + 4\times10^2 + 3\times10^3 + 2\times10^4 + 1\times10^5$ $= 50 + 400 + 3000 + 20000 + 100000$ $= 123450$
Pesquisa em largura primeiro	$r + r^2 + \ldots + r^p + (r^{p+1} - r)$ **Exemplo:** $r = 10$ $p = 5$ Nós gerados $= 10 + 10^2 + 10^3 + 10^4 + 10^5 + (10^{5+1} - 10)$ $= 10 + 100 + 1000 + 10000 + 100000 + 999990$ $= 1111100$

Quadro 2.2: Número total de nós gerados em dois algoritmos de pesquisa

Existe ainda mais um algoritmo de pesquisa, a **pesquisa bidireccional**. Na pesquisa bidireccional realizam-se simultaneamente duas pesquisas. Uma das pesquisas inicia-se no estado inicial e a outra no estado objectivo. A pesquisa bidireccional termina quando estas duas pesquisas se encontrarem no meio. Caso a pesquisa em ambos os sentidos seja feita com recurso à pesquisa em largura primeiro, a pesquisa direccional é completa e óptima, se os custos do passo forem todos idênticos. Através da pesquisa bidireccional consegue-se reduzir tanto a complexidade temporal como a complexidade espacial a $O(r^{p/2})$ [Russ03].

Os vários algoritmos de pesquisa mencionados podem ser sumariados no Quadro 2.3 [Russ03].

Sumário dos algoritmos de pesquisa						
Critério	**Largura primeiro**	**Custo uniforme**	**Prof. primeiro**	**Prof. limitada**	**Aprof. progressivo**	**Bidirecc.**
Completo?	Sim [a]	Sim [a, b]	Não	Não	Sim [a]	Sim [a, d]
Óptimo?	Sim [c]	Sim	Não	Não	Sim [c]	Sim [c, d]
Complexidade temporal	$O\left(r^{p+1}\right)$	$O\left(r^{\lceil c/\min \rceil}\right)$	$O(r^m)$	$O(r^l)$	$O(r^p)$	$O(r^{p/2})$
Complexidade espacial	$O\left(r^{p+1}\right)$	$O\left(r^{\lceil c/\min \rceil}\right)$	$O(rm)$	$O(rl)$	$O(rp)$	$O(r^{p/2})$

(a) Se o factor de ramificação, *r*, for finito.
(b) Se os custos do passo forem superiores ou iguais a um valor mínimo e positivo.
(c) Se o custo do passo for sempre igual.
(d) Se ambas as direcções utilizarem a pesquisa em largura primeiro.

Quadro 2.3: Sumário dos algoritmos de pesquisa

Adaptado de: Stuart Russell e Peter Norvig. Artificial Intelligence – A Modern Approach, Segunda Edição. Prentice Hall, Estados Unidos da América, Upper Saddle River, New Jersey, 2003.

2.3.1.2. Estratégias de pesquisa informadas

As estratégias de pesquisa informadas dispõem de informação adicional acerca do problema, para além daquela que é fornecida na definição desse mesmo problema. Através dessa informação adicional, estas estratégias de pesquisa conseguem chegar mais facilmente à solução do problema do que as estratégias de pesquisa ignorantes.

Uma das estratégias de pesquisa informadas é a **pesquisa sôfrega**. A estratégia de pesquisa sôfrega expande o nó que parece estar mais próximo do objectivo. Para saber qual o nó que parece estar mais próximo do objectivo este algoritmo recorre a uma função heurística, $h(n)$. Esta função é a estimativa do custo do menor caminho desde o nó *n* até ao estado objectivo [Russ03]. Vamos supor que se pretendia encontrar o caminho mais curto entre duas cidades, através de uma rede de cidades intermédias. A função $h(n)$ poderia ser a distância, em linha recta, da cidade *n* até à cidade onde se pretende chegar.

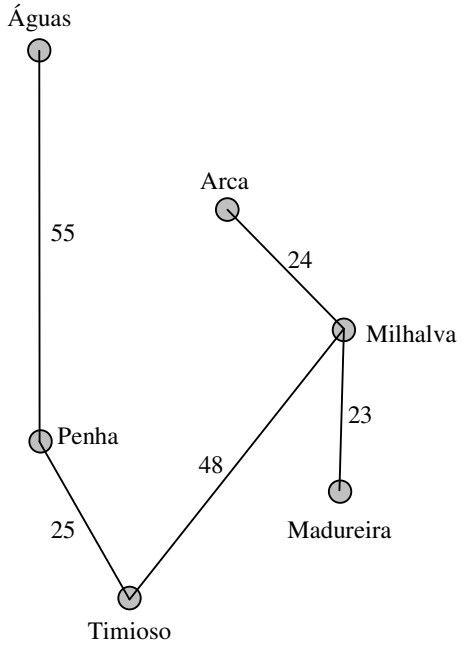


Figura 2.34: Mapa com cidades fictícias

Distância em linha recta entre duas cidades						
Cidades	Águas	Arca	Madureira	Milhalva	Penha	Timioso
Águas	0	34	75	58	55	78
Arca	34	0	42	24	42	56
Madureira	75	42	0	23	43	33
Milhalva	58	24	23	0	45	48
Penha	55	42	43	45	0	25
Timioso	78	56	33	48	25	0

Quadro 2.4: Distância em linha recta entre os locais do mapa da Figura 2.34

Na Figura 2.35 mostra-se a evolução da pesquisa sôfrega, na procura do caminho mais curto da Penha à Madureira. Penha e Madureira são cidades fictícias do mapa da Figura 2.34. Neste mapa, as estradas entre as cidades são representadas por segmentos de recta, junto dos quais é indicada a distância percorrida, em quilómetros. Para $h(n)$, recorre-se à distância, em linha recta, ao objectivo. Neste caso, o objectivo é a Madureira. Como se pode observar, na Figura 2.35, o nó expandido é aquele em que a distância em linha recta, até à Madureira, é menor. As distâncias em linha recta até à Madureira estão por debaixo dos nós. Estes valores foram obtidos através da consulta do Quadro 2.4, onde se encontram as distâncias em linha recta, medidas em quilómetros, entre os vários locais.

Por acaso, no exemplo da Figura 2.35, a pesquisa sôfrega conseguia encontrar a solução óptima. Mas nem sempre se tem esta sorte.

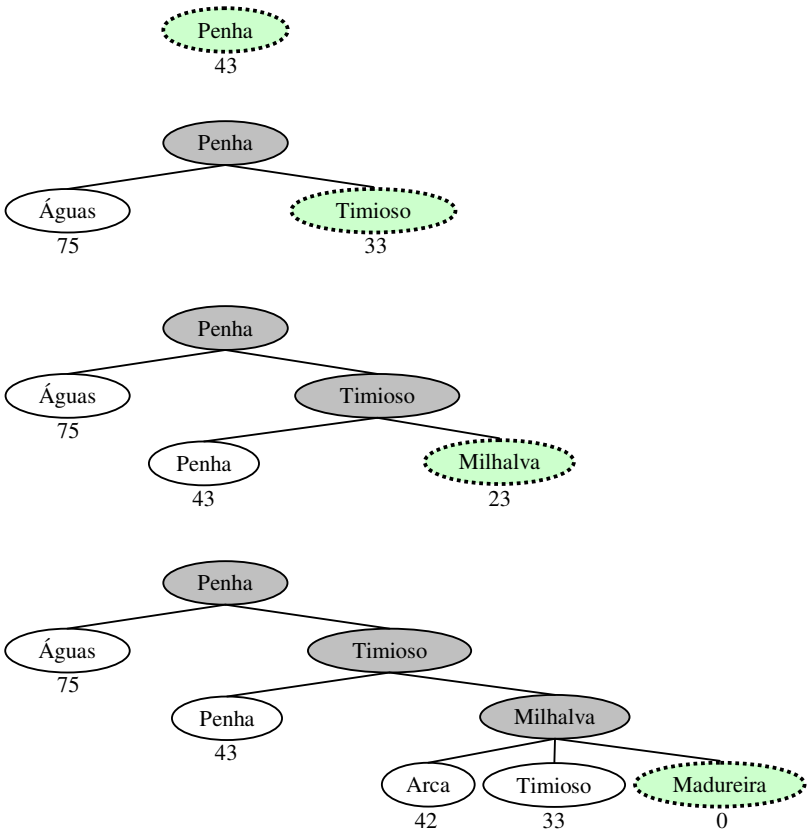


Figura 2.35: Evolução da pesquisa sôfrega

Vamos supor que se pretendia encontrar o caminho das Águas até à Arca. Para saber a distância, em linha recta, de uma determinada cidade até à Arca consulta-se o Quadro 2.4. Na Figura 2.36 mostra-se a evolução da pesquisa sôfrega para este exemplo. Repare-se que o algoritmo fica preso entre Águas e Penha, iterando para sempre entre estes dois locais, nunca chegando a tentar um possível caminho pelo Timioso. A pesquisa sôfrega não é, deste modo, nem completa, nem óptima.

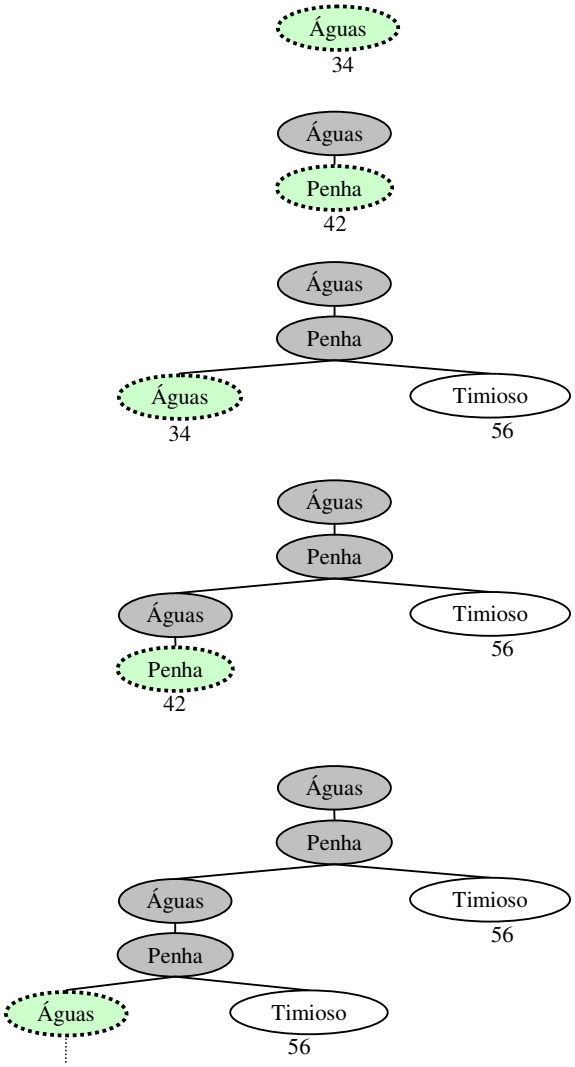


Figura 2.36: Exemplo de pesquisa sôfrega em que nunca se encontra a solução

Outro dos algoritmos de pesquisa informados é a **pesquisa A*** (*A-Star*). Para decidir qual o próximo nó a expandir, a pesquisa A* utiliza a função de avaliação $f(n)$ definida pela Equação (2.15) e em que $c(n)$ é o custo para se atingir o nó *n*. Tal como na pesquisa sôfrega, $h(n)$ é a estimativa do custo do menor caminho, desde o nó *n* até ao objectivo [Nied04, Russ03].

$$f(n) = c(n) + h(n) \tag{2.15}$$

A pesquisa A* é completa, porque nunca fica presa entre dois estados, tal como acontece com a pesquisa sôfrega. Além disso, se $h(n)$ for uma heurística admissível, a pesquisa A* é também óptima. Para que $h(n)$ seja admissível, o custo devolvido por esta função nunca pode

ser superior ao custo do melhor caminho que começa em *n* e acaba no objectivo [Nied04, Russ03].

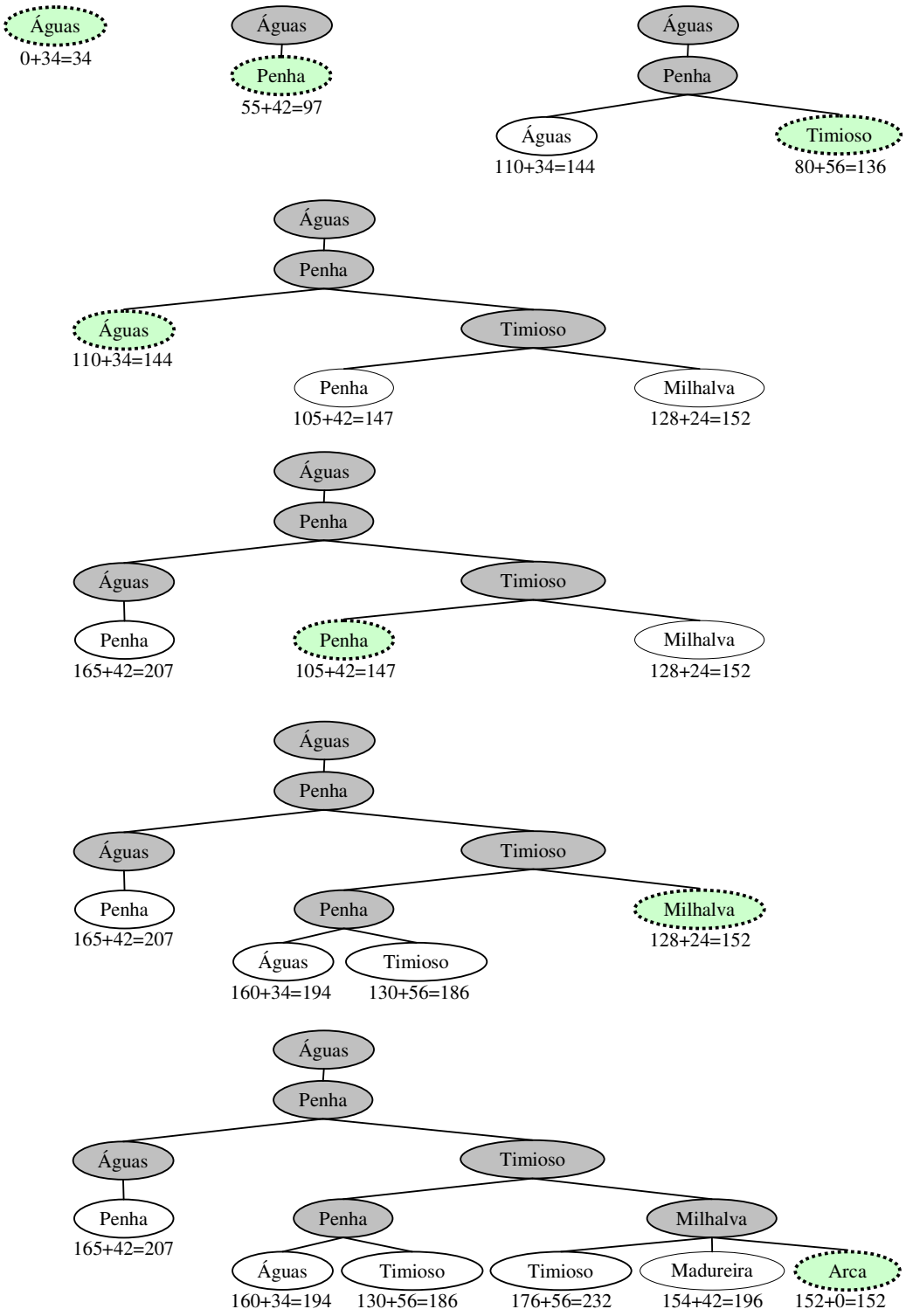


Figura 2.37: Evolução da pesquisa A*

No que se refere ao problema de encontrar o caminho mais curto entre duas cidades, no mapa da Figura 2.34, a distância em linha recta, desde o nó *n* até ao objectivo, é uma heurística admissível. Isto porque o custo do melhor caminho, que vai de *n* ao objectivo, é sempre menor ou igual à distância em linha recta entre *n* e o objectivo.

Na Figura 2.37 mostra-se a evolução da pesquisa A* para encontrar o caminho das Águas até à Arca. Repare-se que, graças à contribuição de $c(n)$, o algoritmo não fica preso entre as Águas e a Penha, tal como acontecia na pesquisa sôfrega. A solução encontrada é a solução óptima.

Existem outros algoritmos, tal com o IDA* ou o SMA* que permitem reduzir a quantidade de memória necessária [Russ03]. Não referimos aqui estes algoritmos porque verificámos, durante a implementação do protótipo, que o A* proporciona bons resultados, não sendo a exigência de memória excessiva. A implementação do protótipo que permite uma visita guiada um museu de pinturas irá ser posteriormente abordada na secção 3.

2.3.2. Geração do caminho de alto nível

O caminho de alto nível é um conjunto de células do modelo ordenadas pela ordem pela qual irá ser efectuada a visita. No caminho de alto nível não é ainda relevante saber qual o percurso que irá ser percorrido dentro das células. Basta saber por que células a visita guiada irá passar.

O caminho de alto nível começa num ponto inicial que é indicado pelo utilizador. Este ponto inicial situa-se dentro de uma determinada célula do modelo. Esta será a primeira célula a incluir no caminho de alto nível. Depois, têm ainda de se adicionar ao caminho de alto nível as células que contêm faces de elevado valor de importância ou aquelas em que o valor de relevância da célula está acima de um determinado limite. Estas são as **células interessantes**.

Há agora que decidir por que ordem estas células devem ser visitadas [Andú04]. A célula correspondente ao ponto indicado pelo utilizador tem de ser a primeira. Depois, há que encontrar o caminho mais curto que permita a visita de todas as células interessantes.

Eventualmente, para chegar a uma célula interessante, poderá ser necessário atravessar uma célula que nada tem de interessante para observar, tal como, por exemplo, um corredor. Não é considerado desvantajoso que isso aconteça. O importante é assegurar que todas as células interessantes sejam visitadas e que o caminho gerado seja o mais curto possível.

Para gerar o caminho de baixo nível pode recorrer-se à pesquisa por aprofundamento progressivo, que é um algoritmo que combina as vantagens da pesquisa em largura primeiro com a pesquisa em profundidade primeiro. Este algoritmo foi descrito na secção 2.3.1.1.

Para efeitos da pesquisa por aprofundamento progressivo considera-se que o custo do passo é sempre 1. Ou seja, passar de uma célula para outra célula tem sempre o custo igual a 1. Sendo o custo do passo constante, a pesquisa por aprofundamento progressivo consegue encontrar a solução óptima, ou seja, o caminho mais curto que passa por todas as células interessantes. O estado objectivo é atingido quando todas as células interessantes estiverem incluídas no caminho.

O grafo de células e portais, cuja criação foi abordada na secção 2.1.3, é utilizado em conjunto com o algoritmo de pesquisa por aprofundamento progressivo, de maneira a saber quais as células que existem no modelo, e quais os portais que permitem a passagem de uma célula para outra.

Se o tempo de execução do algoritmo de pesquisa por aprofundamento progressivo se tornar muito longo, pode-se considerar a hipótese da pesquisa bidireccional, que tem uma complexidade temporal menor. Ou então, poderá simplesmente estabelecer-se um limite máximo para o número de células em que o modelo é dividido. Se este limite for ultrapassado, fundem-se algumas das células [Andú04]. Na secção 2.1.2 foram descritas estratégias para a fusão de células do modelo.

2.3.3. Geração do caminho de baixo nível

A geração do caminho de baixo nível faz-se a seguir à geração do caminho de alto nível. O caminho de alto nível indica por que ordem as células devem ser percorridas. Existe um caminho de baixo nível para cada uma das células do caminho de alto nível. O caminho de baixo nível de uma célula é o caminho que é percorrido dentro dessa célula. O caminho de baixo nível deve evitar atravessar obstáculos, tais como mobília ou paredes.

Na geração do caminho de baixo nível não é ainda relevante saber para onde a câmara está apontada. O caminho de baixo nível é apenas uma sucessão de várias posições através das quais a câmara irá passar.

As várias posições por onde a câmara poderá possivelmente passar são os pontos centrais dos *voxels* vazios no interior da célula. Portanto, cada *voxel* corresponde a uma possível posição da câmara. Sendo assim, constrói-se um grafo em que os vértices são os pontos centrais dos *voxels* e os ramos são a conectividade entre os *voxels* [Andú04].

A conectividade entre os *voxels*, no caminho de baixo nível, determina de que forma a câmara se pode deslocar.

Para ilustrar as possibilidades de movimentação da câmara, consoante a conectividade entre os *voxels*, consideremos a Figura 2.38, onde se encontra o corte horizontal de uma célula de um modelo. Os *voxels* cinzentos são *voxels* não vazios. Neste caso, correspondem às paredes da célula. A célula tem dois portais, um à direita e outro à esquerda. Os *voxels* vazios desta fatia horizontal correspondem aos pontos nos quais a câmara se pode posicionar. Mantendo-se a câmara sempre à mesma altura, considerámos dois tipos possíveis de conectividade entre os *voxels*:

- A câmara pode deslocar-se na horizontal e na vertical.
    - Neste caso, o grafo resultante é o da Figura 2.38 (a).
    - A câmara pode mover-se para uma vizinhança de 4 *voxels*.

- A câmara pode deslocar-se na horizontal, na vertical e também na diagonal.
    - Aqui, o grafo resultante é o da Figura 2.38 (b).
    - A câmara pode mover-se para uma vizinhança de 8 *voxels*.

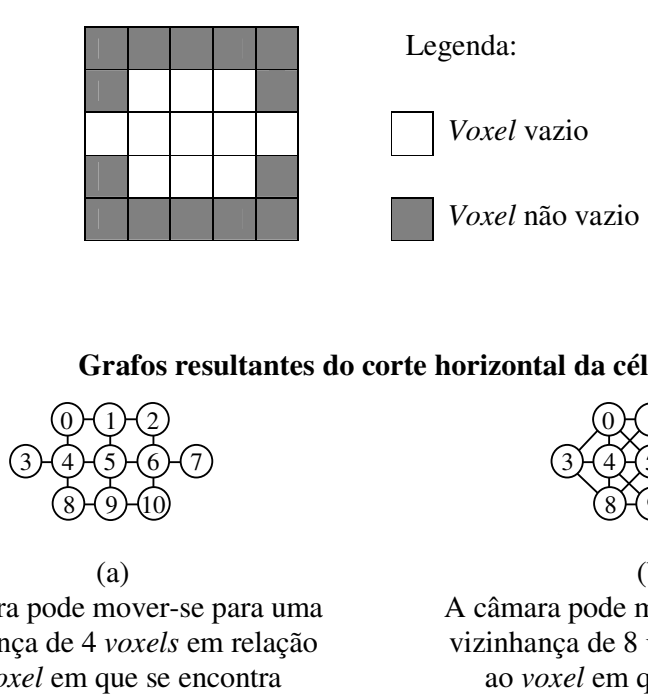


Figura 2.38: Corte horizontal da célula de um modelo e correspondente representação através de um (a) grafo com uma vizinhança de 4 *voxels* e de outro (b) grafo com uma vizinhança de 8 *voxels*

Após se ter criado o grafo que representa a célula, tem de se saber quais as posições pelas quais a câmara irá passar. A primeira posição da câmara corresponde a um dos *voxels* do portal através do qual se entrou na célula. Caso se esteja na primeira célula do caminho de alto nível, a primeira posição da câmara é o ponto definido pelo utilizador.

Depois, encontra-se o caminho mais curto desde este primeiro ponto até ao ponto mais interessante da célula. Se houver mais do que um ponto de interesse na célula, a seguir encontra-se o caminho mais curto desde o primeiro ponto mais interessante até ao segundo ponto mais interessante. Depois, encontra-se o caminho mais curto do segundo ponto mais interessante até ao terceiro ponto mais interessante (se esse ponto existir), e assim sucessivamente, até que não restem mais pontos de interesse dentro da célula. [Andú04]

Quanto se atingir o último ponto de interesse dentro da célula, encontra-se o caminho mais curto desde esse ponto até ao portal que nos levará à próxima célula. Este passo só é necessário caso não se esteja na última célula no caminho de alto nível. Se se estiver na última célula do caminho de alto nível, o caminho de baixo nível termina no último ponto de interesse [Andú04].

Nas células que foram incluídas no caminho de alto nível apenas para possibilitar a passagem entre células interessantes, basta encontrar o caminho mais curto desde o portal de entrada até ao portal de saída.

Na Figura 2.39 encontra-se o corte horizontal de uma célula de um modelo. À semelhança do que acontece na Figura 2.38, também aqui os *voxels* cinzentos são os *voxels* não vazios. Esta célula tem apenas um ponto de interesse, o ponto I. Assinalados a verde encontram-se os vários *voxels* pertencentes ao caminho de baixo nível. Os centros destes *voxels* correspondem

aos pontos pelos quais a câmara irá passar desde a entrada até à saída. A câmara tem 8 graus de liberdade, podendo deslocar-se na vertical, na horizontal e também na diagonal.

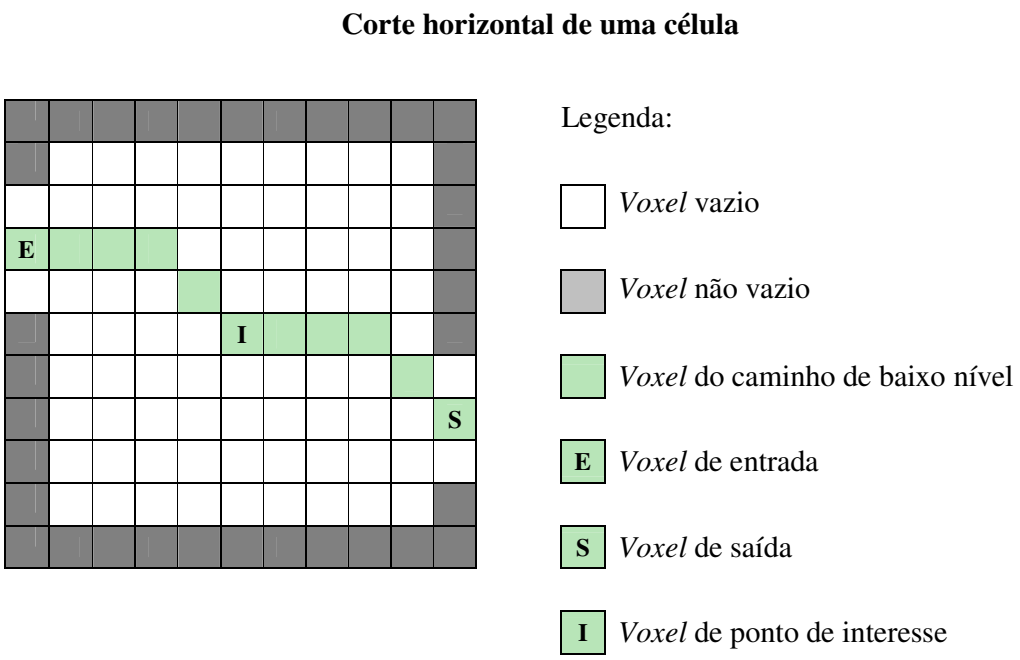


Figura 2.39: *Voxels* por onde passa o caminho de baixo nível de uma célula

Para encontrar o caminho mais curto optou-se pelo algoritmo A*, que já foi anteriormente descrito na secção 2.3.1.2. Para $h(n)$ recorre-se à distância, em linha recta, até ao ponto interessante da célula que se pretende atingir, e que corresponde ao objectivo.

Agora, face à utilização do algoritmo A*, qual será o melhor tipo de conectividade entre os *voxels*? Será bom proporcionar à câmara apenas 4 graus de liberdade? Ou será melhor permitir 8 graus de liberdade?

Vamos supor que a câmara está posicionada no *voxel* assinalado a negro no corte horizontal da grelha de *voxels*, visível na Figura 2.40. A câmara pretende deslocar-se para o *voxel* marcado com uma cruz. Se a câmara tiver 4 graus de liberdade, utilizando a pesquisa A*, é necessário gerar 16 nós, até se encontrar a solução. Os *voxels* por onde a câmara passa são os que têm cor cinzenta. Para este exemplo, proibiu-se a câmara de voltar ao *voxel* de onde acabou de vir. Cada *voxel* tem o comprimento da aresta igual a 1.

Na Figura 2.41 a câmara está, também, inicialmente, posicionada no *voxel* assinalado a negro e pretende deslocar-se para o *voxel* com a cruz. A grelha de *voxels* é a mesma da Figura 2.40. Só que, aqui, vamos permitir à câmara 8 graus de liberdade. Repare-se que, embora o factor de ramificação seja superior ao da Figura 2.40, a solução é encontrada mais rapidamente, sendo gerados apenas 8 nós. Neste exemplo, também se proibiu a câmara de voltar ao *voxel* de onde acabou de vir.

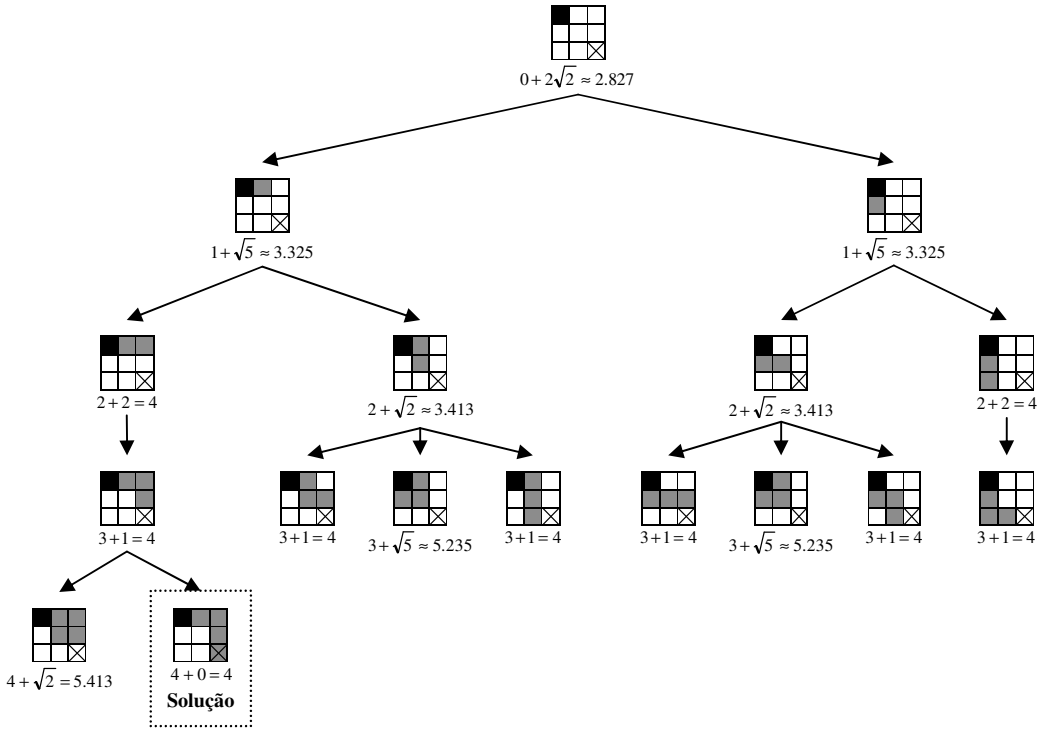


Figura 2.40: Evolução da pesquisa A* desde um *voxel* de origem até um *voxel* de destino, com a câmara a poder mover-se para uma vizinhança de 4 *voxels*

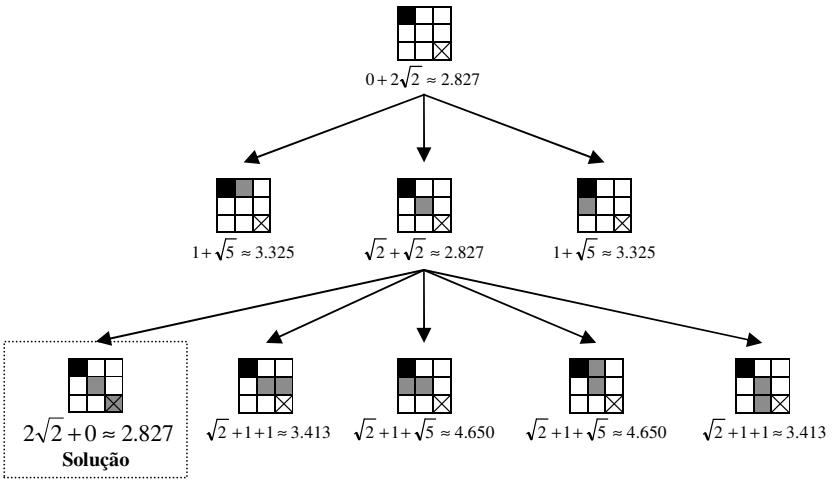
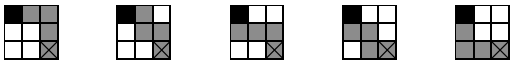


Figura 2.41: Evolução da pesquisa A* desde um *voxel* de origem até um *voxel* de destino, com a câmara a poder mover-se para uma vizinhança de 8 *voxels*

Se se aplicasse a pesquisa A*, com outras grelhas de *voxels*, comparando o número de nós gerados com 4 quatro graus de liberdade e com 8 graus de liberdade da câmara, ir-se-ia chegar a resultados semelhantes aos que se chegou através da Figura 2.40 e da Figura 2.41. Isto acontece porque, com 4 graus de liberdade, o número de soluções óptimas é maior do que com 8 graus de liberdade. Esse número mais elevado de possíveis soluções óptimas leva a pesquisa A* a gerar mais nós.

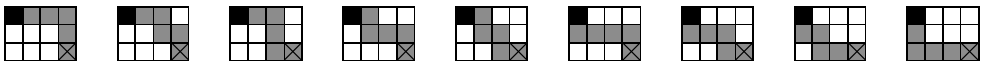
A Figura 2.42 é esclarecedora desta situação. A grelha de voxels desta Figura é igual à da Figura 2.40 e à da Figura 2.41. A origem e o destino da câmara são os mesmos. Com 4 graus de liberdade existem 5 possíveis soluções óptimas. Com 8 graus de liberdade existe apenas 1 solução óptima.



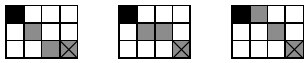
(a)
Soluções óptimas para a pesquisa de um caminho desde um *voxel* de origem até a um *voxel* de destino, tendo a câmara 4 graus de liberdade

(b)
Solução óptima para a pesquisa de um caminho desde um *voxel* de origem até a um *voxel* de destino, tendo a câmara 8 graus de liberdade

Figura 2.42: Soluções óptimas para a pesquisa de um caminho, na fatia de uma grelha de *voxels* 3*3, desde uma origem até um destino, tendo a câmara (a) 4 graus de liberdade e (b) 8 graus de liberdade



(a)
Soluções óptimas para a pesquisa de um caminho desde um *voxel* de origem até a um *voxel* de destino, tendo a câmara 4 graus de liberdade



(b)
Soluções óptimas para a pesquisa de um caminho desde um *voxel* de origem até a um *voxel* de destino, tendo a câmara 8 graus de liberdade

Figura 2.43: Soluções óptimas para a pesquisa de um caminho, na fatia de uma grelha de *voxels* 4*3, desde uma origem até um destino, tendo a câmara (a) 4 graus de liberdade e (b) 8 graus de liberdade

Na Figura 2.43 encontra-se outro exemplo, que demonstra existirem um maior número de soluções caso a câmara tenha 4 graus de liberdade. Com 4 graus de liberdade o número de possíveis soluções óptimas é 9. Com 8 graus de liberdade o número de possíveis soluções óptimas é 3.

Face a tudo o que foi exposto, considera-se que é mais vantajoso proporcionar à câmara 8 graus de liberdade, uma vez que o número de nós gerados até encontrar a solução óptima é menor. Isto leva a que tanto o tempo de execução, como a memória exigida seja menor com 8 graus de liberdade. No entanto, conceder à câmara 4 graus de liberdade é também uma opção praticável, em modelos de menor dimensão.

Uma vez encontrados os pontos pelos quais a câmara irá passar, no caminho de baixo nível, há que construir o caminho propriamente dito. A maneira mais simples de construir o caminho é unir o primeiro ponto do caminho (que corresponde ao *voxel* de entrada) com o segundo ponto do caminho através de um segmento de recta. Depois, une-se o segundo ponto do caminho com o terceiro ponto do caminho através de outro segmento de recta. Seguidamente, une-se o terceiro ponto do caminho com o quarto ponto do caminho através de outro segmento de recta, e assim sucessivamente, até se atingir o último ponto do caminho. A união entre os pontos pode fazer-se através de uma **interpolação linear**. A interpolação linear recorre à Equação (2.16).

$$P = A(1-t) + Bt \tag{2.16}$$

A equação paramétrica apresentada em (2.16) efectua uma interpolação linear entre o ponto A e o ponto B.

Quando $t = 0$, $P = A$. Quando $t = 1$, $P = B$. Fazendo variar t entre 0 e 1 obtêm-se os vários pontos intermédios do segmento de recta que une o ponto A ao ponto B.

Para sabermos a componente x e a componente y podemos recorrer às Equações (2.17). Em três dimensões, teríamos ainda a componente z.

$$\begin{aligned} P_x(t) &= A_x(1-t) + B_x t \\ P_y(t) &= A_y(1-t) + B_y t \end{aligned} \tag{2.17}$$

Para saber as coordenadas dos diversos pontos entre A e B pode-se recorrer à Equação (2.18), que congrega as duas Equações apresentadas em (2.17).

$$(x(t), y(t)) = \left(A_x(1-t) + B_x t, A_y(1-t) + B_y t\right) \tag{2.18}$$

A título de exemplo, no Quadro 2.5 encontram-se as coordenadas dos diversos pontos intermédios para uma interpolação linear entre o ponto $A(0,1)$ e o ponto $B(2,4)$.

Pontos a interpolar	Coordenadas dos pontos	
	X	*Y*
Ponto *A*	0	1
Ponto *B*	2	4

Coordenadas dos pontos intermédios resultantes da interpolação linear entre *A* e *B*			
Pontos	***t***	***X***	***Y***
A	0	0	1
Pontos intermédios	0,1	0,2	1,3
	0,2	0,4	1,6
	0,3	0,6	1,9
	0,4	0,8	2,2
	0,5	1	2,5
	0,6	1,2	2,8
	0,7	1,4	3,1
	0,8	1,6	3,4
	0,9	1,8	3,7
B	1	2	4

Quadro 2.5: Coordenadas dos pontos intermédios resultantes da interpolação linear entre dois pontos

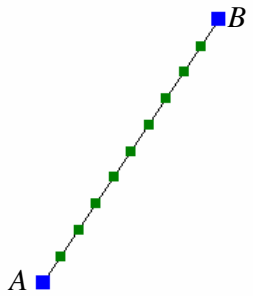


Figura 2.44: Interpolação linear entre dois pontos

Para percorrer a distância de *A* a *B* a câmara posicionar-se-ia nos vários pontos intermédios, assinalados a verde na Figura 2.44, transmitindo desse modo a ideia de movimento. A diferença entre um valor de *t* e o seguinte é de 0,1. Claro, podia-se escolher um valor *t* mais pequeno, caso se achasse que o movimento da câmara não era suficientemente fluido.

A interpolação linear, do ponto de vista matemático, é extremamente simples e fácil de implementar. Além disso, o tempo de execução é também reduzido, uma vez que os cálculos são simples. No entanto, o caminho de baixo nível obtido através deste método não é suave, o que poderá causar algum desconforto e estranheza ao utilizador. A visita guiada será mais agradável caso o caminho seja curvilíneo e suave, sem transições bruscas entre os diversos pontos. Para tal, é necessário, criar uma curva.

2.3.4. Suavização do caminho de baixo nível através de uma curva

As curvas têm muitas aplicações. Devido à sua estética, os designers utilizam-nas nos seus modelos. Outros, utilizam-nas para representar um objecto da natureza. Um engenheiro poderá utilizá-las na concepção de um novo avião ou automóvel. As curvas podem também descrever a trajectória de um objecto numa animação. No caso específico desta tese, as curvas são úteis para efeitos da descrição da trajectória da câmara.

Como já se referiu, a curva deverá ser suave. É conveniente definir-se aqui o que se entende por "suave" para que o conceito não seja apenas qualitativo. Uma curva é suave se tiver **continuidade C^k**. Também se pode denominar a continuidade C^k por **continuidade paramétrica de ordem k**. Uma curva tem continuidade C^k, no intervalo t [a, b], se todas as derivadas da curva, até à ordem k, existirem e forem contínuas para todos os pontos no intervalo [a, b] [Fole00, Hear97, Hill01, Sant06].

Para que uma curva tenha continuidade C^0, basta que essa curva seja contínua [Fole00, Hear97, Hill01]. A curva da esquerda da Figura 2.45 é uma curva contínua. A curva é constituída por duas secções, e a primeira secção termina no ponto onde a segunda secção começa. Isso basta para que a curva seja contínua. No entanto, esta curva não é suave, ou seja não tem continuidade C^1, uma vez que no ponto *A* a derivada de ordem 1 da primeira secção da curva não é igual à derivada de ordem 1 da segunda secção da curva.

A curva da direita da Figura 2.45 tem continuidade C^0 e C^1. Esta curva é também constituída por duas secções, que se encontram no ponto *A*. Neste ponto, as derivadas de ordem 1 de ambas as secções são iguais, pelo que a curva é suave.

Curva com continuidade C^0, mas sem continuidade C^1

Curva com continuidade C^0 e com continuidade C^1

Figura 2.45: Continuidade C^0 e C^1

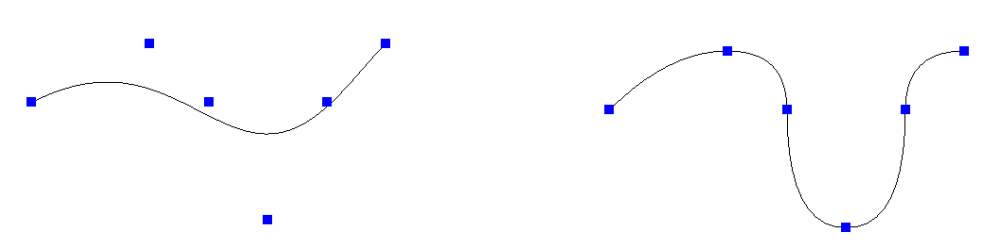
Numa curva com continuidade C^2, tanto a primeira como a segunda derivada das duas secções da curva são iguais no ponto de intersecção. Numa curva paramétrica deste tipo, e interpretando-se o parâmetro como tempo, não existem mudanças abruptas de aceleração na transição entre duas secções da curva [Fole00, Hear97, Hill01].

Relativamente à geração da curva para o caminho de baixo nível, o óptimo será beneficiar de continuidade C^2. Dessa forma, a deslocação da câmara será o mais suave possível.

No entanto, não basta que a curva correspondente ao caminho de baixo nível seja suave. Essa curva deverá também ser construída com base nos pontos pelos quais a câmara passa. Existem vários métodos para construir uma curva com base num conjunto de pontos. Esses métodos podem-se agrupar em duas grandes categorias [Hear97, Hill01]:

- Geração de uma curva por aproximação
- Geração de uma curva por interpolação

Na geração de uma curva por aproximação, a curva aproxima-se dos pontos. No entanto, a curva não tem obrigatoriamente de passar por esses pontos. Na geração de uma curva por interpolação, a curva passa obrigatoriamente por todos os pontos.



Curva gerada por aproximação Curva gerada por interpolação

Figura 2.46: Aproximação e interpolação de curvas

Na Figura 2.46, à direita, encontra-se uma curva gerada por aproximação e, à esquerda, uma curva gerada por interpolação. Os pontos nos quais se baseia o desenho da curva estão assinalados através de quadrados.

Entre as curvas geradas por aproximação encontram-se as curvas de Bézier e as curvas B-Spline. No se refere à geração de uma curva por interpolação, as curvas de Hermite são muito populares. Nas secções seguintes ir-se-á analisar estas curvas no que se refere à sua utilidade para a geração da curva do caminho de baixo nível.

2.3.4.1. Curvas de Bézier

As curvas de Bézier foram desenvolvidas por Pierre Bézier, por volta de 1962, sendo utilizadas no design dos automóveis da marca Renault [Chas78, Fole00, Hear97, Hill01]. As curvas de Bézier são um tipo de curvas geradas por aproximação. As curvas de Bézier podem ser desenhadas recorrendo ao **algoritmo de De Casteljau**. O nome do algoritmo de De Casteljau é uma homenagem a Paul De Casteljau, que também deu um contributo muito valioso para a criação deste tipo de curva [Hill01].

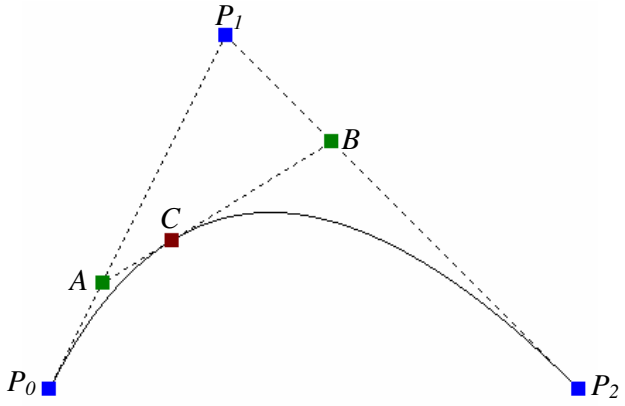
O algoritmo De Casteljau recebe um conjunto de pontos em se irá basear o desenho da curva. Esses pontos podem ser denominados por **pontos de controlo** [Hear97, Hill01]. Por uma

questão de simplicidade, vamos supor que recebemos três pontos de controlo P_0, P_1 e P_2. Para a geração de qualquer dos pontos da curva, procede-se da seguinte maneira [Burg89, Fole00, Hear97, Hill01, Sant06]:

1. Escolhe-se um valor de *t* entre 0 e 1.
2. Encontra-se o ponto *A*, que está a uma fracção *t* do caminho entre P_0 e P_1. Pode-se encontrar o ponto *A* recorrendo à Equação da interpolação linear, já anteriormente apresentada. Ou seja, $A = P_0(1-t) + P_1 t$.
3. Encontra-se o ponto *B*, que está a uma fracção *t* do caminho entre P_1 e P_2. Novamente, pode-se recorrer à interpolação linear. Desta vez, $B = P_1(1-t) + P_2 t$.
4. Finalmente, encontra-se o ponto *C*, que está a uma fracção *t* do caminho entre *A* e *B*. Neste caso, $C = A(1-t) + Bt$.

Repetindo este procedimento para vários valores de *t*, encontram-se os vários pontos da curva de Bézier. A curva começa em P_0 e acaba em P_2. Para desenhar a curva une-se P_0 ao ponto que corresponde ao valor mais baixo de *t*. Depois une-se o valor mais baixo de *t* ao segundo valor mais baixo de *t*, e assim sucessivamente até se chegar a P_2. Quantos mais valores de *t* se escolherem, mais fluida será a curva. Normalmente, escolhem-se valores de *t* a intervalos regulares [Burg89, Hear97, Hill01].

Na Figura 2.47 encontra-se a geração de um ponto de uma curva de Bézier com $t = 0{,}3$. A curva tem como pontos de controlo $P_0(0, 0)$, $P_1(2, 4)$ e $P_2(6, 0)$. Após se efectuar os cálculos encontram-se os pontos A, B e, finalmente, o ponto *C*, que é um dos pontos da curva. O ponto *C* tem as coordenadas (1.38, 1.68).



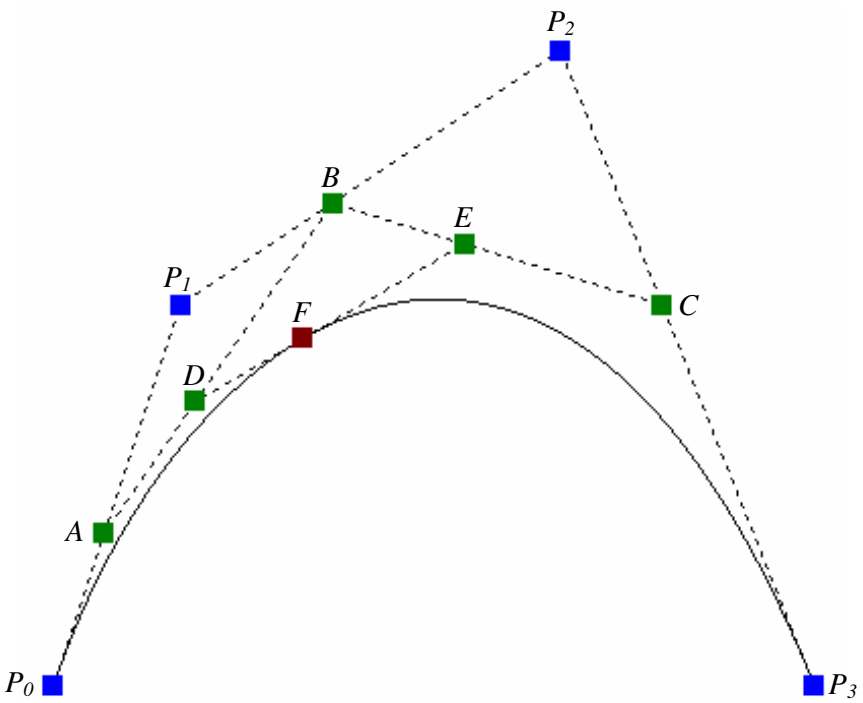
Pontos de controlo da curva
P_0 (0, 0)
P_1 (2, 4)
P_2 (6, 0)

$t = 0.3$

A (0.60, 1.20)
B (3.20, 2.80)
C (1.38, 1.68)

C pertence à curva de Bézier.

Figura 2.47: Curva de Bézier com 3 pontos de controlo



Pontos de controlo da curva
P_0 (0, 0)
P_1 (1, 3)
P_2 (4, 5)
P_3 (6, 0)

t = 0.4

A(0.400, 1.200)
B(2.200, 3.800)
C(4.800, 3.000)
D(1.120, 2.240)
E(3.240, 3.480)
F(1.968, 2.736)

F pertence à curva de Bézier.

Figura 2.48: Curva de Bézier com 4 pontos

Este processo pode ser aplicado a um maior número de pontos [Burg89, Hear97, Hill01]. Na Figura 2.48 encontra-se a geração de um ponto de uma curva de Bézier baseada em quatro pontos, P_0(0, 0), P_1(1, 3), P_2(4, 5) e P_3(6, 0), com t igual a 0,4. Como se pode observar, começa-se por encontrar o ponto A, que está a uma fracção de 0,4 do caminho entre P_0 e P_1. Para encontrar os pontos B e C procede-se de uma forma semelhante. Depois, encontra-se o ponto D que está a uma fracção 0,4 do caminho entre A e B. Da mesma forma, o ponto E está a uma fracção 0,4 do caminho entre B e C. Finalmente, temos o ponto F, pertencente à curva de Bézier, que está a uma fracção 0,4 do caminho entre D e E. O ponto F tem as coordenadas (1.968, 2.736).

Uma curva de Bézier pode ser descrita com recurso à Equação (2.19) e aos seus polinómios de Bernstein, dados pelas Equações (2.20) e (2.21) [Burg89, Hear97, Hill01, Sant06, Shre04].

$$P(t) = \sum_{k=0}^{L} P_k B_k^L(t) \tag{2.19}$$

$$B_k^L(t) = \binom{L}{K}(1-t)^{L-k} t^k \tag{2.20}$$

$$\binom{L}{K} = \frac{L!}{k!(L-k)!}, \text{ para } L \geq k \tag{2.21}$$

$B_k^L(t)$ é polinómio de Bernstein de grau k. A curva baseia-se em $L + 1$ pontos de controlo. Os pontos de controlo são, deste modo, os pontos P_0, P_1, até P_L. Fazendo variar t entre 0 e 1 obtém-se os vários pontos da curva de Bézier [Hear97, Hill01].

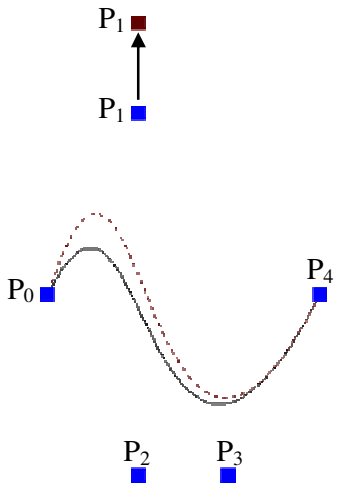


Figura 2.49: Falta de controlo local em curva de Bézier

Um dos problemas das curvas de Bézier é a sua falta de controlo local. Ou seja, quando se desloca um dos pontos em que a curva é baseada, toda a curva é afectada [Fole00, Hear97, Hill01]. Na Figura 2.49 encontra-se uma curva de Bézier, com cinco pontos de controlo, desenhada a preto. A alteração da posição do ponto P1, como se pode observar, produziu alterações ao longo de toda a curva. A nova curva é desenhada a traço interrompido vermelho escuro.

A falta de controlo local pode revelar-se um problema se a curva de Bézier for utilizada para o design de um novo objecto em que seja importante para o criador ter um grande controlo sobre as formas geradas. No entanto, no que se refere ao caminho de baixo nível, a forma da curva não assume uma importância tão elevada. O que é importante é que a curva seja suave e que proporcione ao utilizador uma deslocação agradável.

Há também que ter em conta que a curva de Bézier é uma curva gerada por aproximação a um conjunto de pontos. Lembramos que o caminho de baixo nível é construído com base num conjunto de pontos. Existindo uma sequência de pontos que nos leva desde a entrada da célula até ao primeiro ponto de interesse, podemos construir um curva de Bézier que começa no ponto inicial desse caminho e que acaba no primeiro ponto de interesse da célula. Uma vez que as curvas de Bézier aproximam os pontos, sem forçosamente os interpolar, eventualmente, o caminho de baixo nível gerado poderá afastar-se demasiado, acabando eventualmente por chocar com algum obstáculo arquitectónico. É certo que esta situação é pouco provável, uma vez que os pontos que servirão de base ao caminho de baixo nível são bastante densos, puxando a curva ao seu encontro e não deixando muita margem para esta se afastar.

Além disso, a curva de Bézier está limitada ao seu Envoltório Convexo (*Convex Hull*), tal como se pode ver na Figura 2.50. Em duas dimensões, pode-se conceber o Envoltório Convexo como um elástico que é colocado à volta dos pontos, tal como se mostra na Figura 2.50 [Fole00, Hear97, Hill01]. Portanto, por mais que a curva se possa eventualmente afastar, ela nunca ultrapassará os limites do seu Envoltório Convexo. Sendo assim, se o interior do Envoltório Convexo não intersectar nenhum obstáculo arquitectónico é garantido que a curva do caminho de baixo nível também não o intersectará.

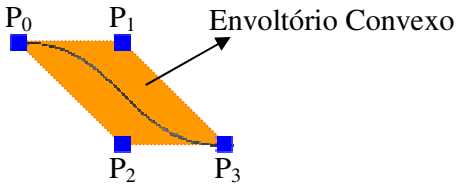


Figura 2.50: Envoltório Convexo de uma curva de Bézier

No entanto, se ainda assim se considerar necessário "prender" de forma mais disciplinada a curva aos pontos que servem de base ao caminho de baixo nível, pode-se construir, em vez de apenas uma única curva de Bézier, várias curvas de Bézier. Então, no caso do conjunto de pontos, pertencentes ao caminho de baixo nível, que vão desde o portal até ao primeiro ponto de interesse pode-se criar uma curva de Bézier a, por exemplo, cada quatro pontos. Da união dessas curvas resultará uma curva maior. Uma vez que as curvas de Bézier passam sempre pelo primeiro ponto e pelo último ponto da curva, a continuidade C^0 está assegurada. No entanto, a continuidade C^0 não é suficiente para garantir uma suave deslocação da câmara [Fole00, Hear97, Hill01].

Para unir duas curvas de Bézier, garantindo a continuidade C^1, é necessário que o ponto de controlo P_{n-1} e o ponto de controlo P_n, da primeira curva, sejam colineares com o ponto de controlo P_0 e o ponto de controlo P_1, da segunda curva. É também forçoso que a distância de P_{n-1} a P_n seja igual à distância de P_0 a P_1 [Chas78, Fole00, Hear97].

Na Figura 2.51 encontram-se duas curvas de Bézier unidas como continuidade C^1. Como se pode observar P_{2A}, P_{3A}, P_{0B} e P_{1B} estão posicionados ao longo da mesma linha. Além disso, $\overline{P_{2A}P_{3A}} = \overline{P_{0B}P_{1B}}$.

Para que se consiga continuidade C^1, no caminho de baixo nível, poderá ser necessária a geração de alguns pontos adicionais para esse mesmo caminho, de forma a obter pontos colineares.

É possível forçar que a união de duas curvas de Bézier tenha continuidade C^2. Só que, para que tal aconteça é necessário impor limitações muito restritivas ao posicionamento dos pontos de controlo, o que dificultaria em demasia a geração do caminho de baixo nível. [Hear97].

Na secção seguinte é abordado outro tipo de curva, a curva B-Spline.

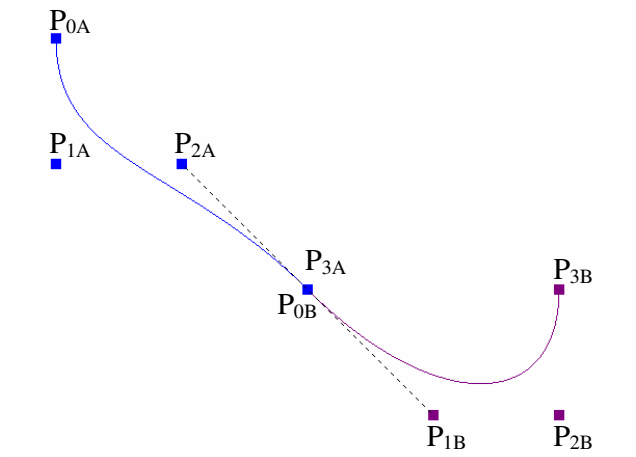


Figura 2.51: Duas curvas de Bézier unidas com continuidade C^1

2.3.4.2. Curvas B-Spline

As curvas B-Spline, à semelhança das curvas de Bézier, permitem aproximar um conjunto de pontos. Só que, enquanto nas curvas de Bézier a modificação da posição de um dos pontos de controlo da curva afecta a totalidade da curva, nas curvas B-Spline, a modificação da posição de um dos pontos de controlo da curva afecta a curva apenas na proximidade desse ponto. As curvas B-Spline têm, dessa forma, maior controlo local [Hear97, Hill01, Sant06]. Esta vantagem é especialmente importante no design de um novo objecto em que todos os pormenores sejam importantes. No entanto, na geração do caminho de baixo nível de uma célula, esta vantagem não é tão relevante.

Para gerar uma curva B-Spline recorre-se às Equações (2.22), (2.23) e (2.24) [Hear97, Hill01].

$$P(t) = \sum_{k=0}^{L} P_k N_{k,m}(t) \tag{2.22}$$

$$N_{k,m}(t) = \left(\frac{t - t_k}{t_{k+m-1} - t_k} \right) N_{k,m-1}(t) + \left(\frac{t_{k+m} - t}{t_{k+m} - t_{k+1}} \right) N_{k+1,m-1}(t) \tag{2.23}$$

$$N_{k,1}(t) = \begin{cases} 1 & \text{se } t_k < t \le t_{k+1} \\ 0 & \text{caso contrário} \end{cases} \tag{2.24}$$

À semelhança do que acontece com as curvas de Bézier, as curvas B-Spline baseiam-se em $L + 1$ pontos de controlo. Os pontos de controlo são os pontos P_0, P_1 até P_L. $N_{k,m}(t)$ é a fórmula geral das funções B-Spline. Como se pode observar, estas funções são definidas de uma forma recursiva. $N_{k,1}(t)$ é a função B-Spline de primeira ordem.

Fazendo variar t entre zero e um obtêm-se os vários pontos da curva B-Spline, à semelhança do que acontece nas curvas de Bézier.

A ordem das funções B-Spline é indicada pelo parâmetro m, não podendo m exceder o número de pontos de controlo. De um modo geral, m é igual a 3 ou a 4. Se m for igual a 3, temos um curva B-Spline quadrática. Se m for igual a 4, a curva B-Spline é cúbica.

As curvas B-Spline recorrem ainda a um vector de nós (*knot vector*), sendo que o vector de nós é $T = (t_0, t_1, t_2, \ldots)$ [Hear97, Hill01].

Repare-se que, nas curvas B-Spline, é possível aumentar o número de pontos de controlo sem aumentar a ordem das funções B-Spline. Isto é uma vantagem em relação às curvas de Bézier, em que um maior número de pontos de controlo acarreta um aumento da ordem dos polinómios de Bernstein [Hear97, Hill01].

As curvas B-Spline têm continuidade C^{m-2}, o que permite controlar a suavidade da curva gerada [Hear97, Hill01]. Esta é uma propriedade importante no que se refere à geração do caminho de baixo nível, uma vez que, para efeitos do conforto do utilizador é relevante que a curva gerada seja o mais suave possível.

Um tipo de curva B-Spline especialmente interessante para a geração do caminho de baixo nível é a curva B-Spline Aberta (*Open B-Spline*), porque, através deste tipo de curvas é possível forçar a interpolação do primeiro ponto de controlo e do último ponto de controlo. Nas curvas B-Spline Abertas o espaçamento entre os nós é uniforme, à excepção do início e do fim onde os nós são repetidos *m* vezes. Sendo assim, para uma Curva B-Spline Aberta com L+1 pontos de controlo e ordem m, o vector de nós tem as seguintes características [Hear97, Hill01]:

- Existem $L + m + 1$ nós.
- Os nós vão de t_0 até t_{L+m}.
- Os primeiros *m* nós, que vão de t_0 até t_{m-1}, são iguais a 0.
- t_m é igual a 1.
- Os nós de t_m até t_L sofrem incrementos no valor de 1.
- t_L é igual a $L - m + 1$.
- Os últimos *m* nós, que vão de t_{L+1} até t_{L+m} são iguais a $L - m + 2$.

No Quadro 2.6 encontram-se alguns exemplos do vector de nós para curvas B-Spline Abertas com diferentes números de pontos de controlo e diferentes valores de *m*.

É curioso verificar que as curvas de Bézier são, na realidade, um caso especial das curvas B-spline. Quando $m = L + 1$, a curva B-Spline Aberta transforma-se numa curva de Bézier. Ou seja, as funções B-Spline convertem-se nos polinómios de Bernstien. Nesse caso, todos os nós serão iguais a 0 ou iguais a 1. Mais propriamente, os primeiros *m* nós são todos iguais a 0 e os últimos *m* nós são todos iguais a 1 [Hear97, Hill01].

Vector de nós para curvas B-Spline Abertas				
Nº de pontos de controlo *(L+1)*	Nº de pontos de controlo menos 1 *(L)*	Ordem das funções B-Spline (*m*)	Número de nós (*L* + *m* + 1)	Vector de nós (*T*)
4	3	2	6	$T = (0,0,1,2,3,3)$
5	4	4	9	$T = (0,0,0,0,1,2,2,2,2)$
8	7	4	12	$T = (0,0,0,0,1,2,3,4,5,5,5,5)$
6	5	6	12	$T = (0,0,0,0,0,0,1,1,1,1,1,1)$

Quadro 2.6: Vector de nós para curvas B-Spline Abertas

No último exemplo do Quadro 2.6 encontra-se uma curva B-Spline Aberta que, na realidade, é uma curva Bézier. Repare-se que os primeiros seis nós são todos iguais a 0 e os últimos seis nós são todos iguais a 1.

Na realidade, à medida que a ordem das funções B-Spline aumenta, diminui o controlo local. Quando *m* atinge o limite de *L*+1 o controlo local é mínimo e a modificação da posição de um ponto de controlo afectará a forma da totalidade de curva [Hear97, Hill01].

Na Figura 2.52 encontram-se três curvas B-Spline Abertas. Cada ponto de controlo é representado por um quadrado. Todas as curvas têm catorze pontos de controlo. Apenas o parâmetro *m* se altera. Como se pode ver, à medida que *m* aumenta, a curva resultante fica menos "presa" aos pontos de controlo. A curva em que *m* é igual a catorze é também uma curva Bézier.

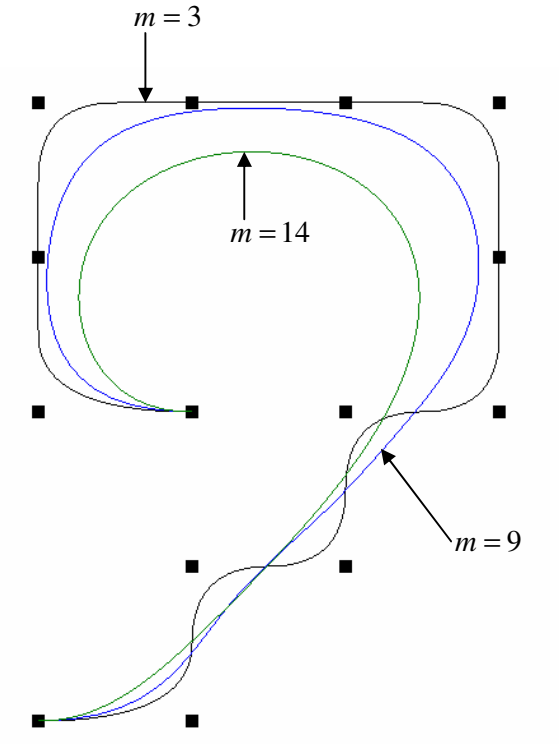


Figura 2.52: Curvas B-Spline Abertas com os mesmos pontos de controlo mas com diferentes ordens das funções B-Spline

Outro tipo de curvas são as Beta-Splines ou β-Splines. Neste tipo de curva não é simples assegurar a continuidade paramétrica [Hear97, Hill01, Sant06], pelo que não iremos aqui descrevê-las em maior pormenor.

Por tudo o que foi referido nesta secção, parece-nos que as curvas B-Spline Abertas são uma boa opção para a geração do caminho de baixo nível, uma vez que permitem a aproximação de uma série de pontos de controlo que corresponderão aos vários pontos do caminho de baixo nível. Além disso, é possível manter a suavidade da curva, uma vez que é garantido que as curvas B-Spline têm sempre continuidade C^{m-2}.

No entanto, existe ainda mais uma forma de poder gerar a curva para o caminho de baixo nível: através da interpolação. Recordamos que tanto as curvas de Bézier como as curvas B-Spline aproximam pontos. Numa interpolação, a curva gerada passa por todos os pontos. Na secção seguinte iremos abordar a interpolação de Hermite.

2.3.4.3. Interpolação de Hermite

Na interpolação, a curva passa obrigatoriamente por todos os pontos de controlo. Sendo assim, vamos supor que temos vários pontos de controlo P_0, P_1, P_2, até P_L. Pretende-se construir uma curva suave que passe por todos estes pontos. Se tivermos cinco pontos de controlo a curva irá ser constituída por cinco secções. A primeira secção é uma curva que liga o ponto P_0 ao ponto P_1. A segunda secção é outra curva que liga o ponto P_1 ao ponto P_2. A terceira secção liga P_2 a P_3 e por aí adiante, até que, finalmente, a último secção une o ponto P_4, com o ponto P_5. Cada uma destas secções é um polinómio cúbico. A Equação destes polinómios é:

$$R_k(t) = A_k t^3 + B_k t^2 + C_k t + D_k \tag{2.25}$$

$$\text{Nº de pontos de controlo da curva} = L+1$$

$$k = 0,1,\ldots,L-1$$

Note-se que k varia entre zero e o número de pontos de controlo da curva menos dois. Também se pode dizer que k varia entre zero e o número de secções da curva menos um.

Se a curva tiver cinco pontos de controlo $R_0(t)$ é a primeira secção da curva, que o une o ponto P_0, ao ponto P_1. Por sua vez, $R_3(t)$ será a última secção da curva que une o ponto P_3 ao ponto P_4.

Os coeficientes são A_k, B_k, C_k e D_k.

O parâmetro t varia entre zero e um para cada uma das secções da curva. Repare-se que, nas curvas B-Spline, fazendo variar t entre zero e um, na Equação (2.22), consegue-se desenhar toda a curva. Na interpolação de Hermite, a curva total desenha-se por secções, fazendo variar t entre zero e um para cada uma das secções.

Para cada k existe uma componente em x, uma componente em y e, eventualmente, também uma componente em z. Em duas dimensões, as componentes para a secção k podem ser descritas pela Equações indicadas em (2.26).

$$x_k(t) = ax_k t^3 + bx_k t^2 + cx_k t + dx_k$$
$$y_k(t) = ay_k t^3 + by_k t^2 + cy_k t + dy_k \quad (2.26)$$

$x_k(t)$ é a componente *x* de $R_k(t)$.

ax_k, bx_k, cx_k e dx_k são os coeficientes para a componente *x*.

$y_k(t)$ é a componente *y* de $R_k(t)$.

ay_k, by_k, cy_k e dy_k são os coeficientes para a componente *y*.

t varia entre zero e um.

Falta apenas, agora, determinar quais são os coeficientes. De acordo com a interpolação de Hermite, os coeficientes são obtidos através das Equações que se encontram em (2.27). Caso a coordenada *z* fosse relevante, procedia-se de forma semelhante para a componente *z*.

$$ax_k = Sx_{k+1} + Sx_k - 2(x_{k+1} - x_k)$$
$$bx_k = 3(x_{k+1} - x_k) - 2Sx_k - Sx_{k+1}$$
$$cx_k t = Sx_k$$
$$dx_k = x_k$$
$$ay_k = Sy_{k+1} + Sy_k - 2(y_{k+1} - y_k)$$
$$by_k = 3(y_{k+1} - y_k) - 2Sy_k - Sy_{k+1}$$
$$cy_k t = Sy_k$$
$$dy_k = y_k \quad (2.27)$$

x_k é a coordenada *x* do ponto P_k.

y_k é a coordenada *y* do ponto P_k.

x_{k+1} é a coordenada *x* do ponto P_{k+1}.

y_{k+1} é a coordenada *y* do ponto P_{k+1}.

Sx_k e Sy_k são as derivadas paramétricas no ponto P_k.

Sx_{k+1} e Sy_{k+1} são as derivadas paramétricas no ponto P_{k+1}.

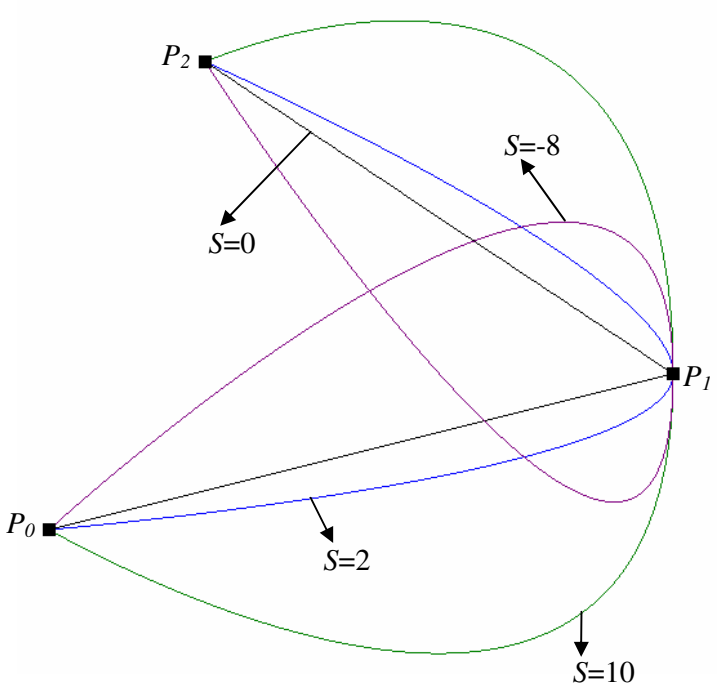
Também se pode denominar Sx_k, Sy_k, Sx_{k+1} e Sy_{k+1} por vectores tangentes, sendo valores fornecidos pelo utilizador.

Através das Equações de (2.27) é possível unir o ponto P_k ao ponto P_{k+1} através de uma curva.

Recorrendo à utilização das Equações mencionadas nesta secção é possível unir pontos através de curvas Hermite. Como exemplo, vamos supor que pretendemos realizar uma interpolação de Hermite entre os pontos $P_0(0,0)$, $P_1(4,1)$ e $P_2(1,3)$.

No que se refere à secção da curva entre P_0 e P_1, a derivada paramétrica quando t é igual a zero é (0, 0). Quando t é igual a 1, a derivada paramétrica é (0, S). No que se refere à secção da curva entre P_1 e P_2, a derivada paramétrica quando t é igual a zero é (0, S). Quando t é igual a 1, a derivada paramétrica é (0, 0).

Na Figura 2.53 encontra-se o desenho da curva que interpola estes três pontos, para diferentes valores de S. Repare-se que, modificando os valores de S, é possível produzir diferentes curvas, todas com continuidade C^1, uma vez que as derivadas de primeira ordem são iguais no ponto de junção das duas secções.



Curva Hermite com três pontos de controlo	
Pontos de controlo da curva	**Derivada no ponto**
$P_0(0,0)$	$Sx_0 = 0$ $Sy_0 = 0$
$P_1(4,1)$	$Sx_1 = 0$ $Sy_1 = S$
$P_2(1,3)$	$Sx_2 = 0$ $Sy_2 = 0$

Figura 2.53: Curva Hermite com três pontos de controlo

As Equações mencionadas nesta secção permitem interpolar pontos. No entanto, fornecer os valores de Sx_k, Sy_k, Sx_{k+1} e Sy_{k+1} pode ser complicado para o utilizador. As curvas Catmull-Rom são baseadas na interpolação de Hermite e permitem estabelecer as derivadas nos pontos de controlo (Sx_k, Sy_k, Sx_{k+1} e Sy_{k+1}) de forma automática.

De acordo com o método Catmull-Rom, a derivada no ponto de controlo P_k vai ser um valor $P'(t_k)$, baseado na posição dos dois pontos vizinhos a P_k na curva. Para calcular $P'(t_k)$ utiliza-se a Equação (2.28). Esta Equação tem uma componente *x* e uma componente *y*, e caso seja necessário, também uma *z*.

$$P'(t_k) = v(P_{k+1} - P_{k-1}) \tag{2.28}$$

P_{k-1} é o ponto anterior a P_k na curva.

P_{k+1} é o ponto seguinte a P_k na curva.

v é um valor que pode ser escolhido pelo utilizador.

Portanto, para a determinação de $P'(t_k)$ num ponto de controlo são necessários, no total, três pontos de controlo.

Os pontos de controlo intermédios de uma curva têm todos um ponto de controlo anterior e um ponto de controlo seguinte. Mas, o que fazer com o primeiro ponto de controlo e com o último ponto de controlo? O primeiro ponto de controlo não tem nenhum antes dele. E o último ponto de controlo não tem nenhum depois dele.

Uma possível solução é fazer com que as segundas derivadas nos pontos P_0 e P_L sejam iguais a zero.

Outra solução consiste em acrescentar dois pontos fantasma à curva, os pontos $P_{-1}(x_{-1}, y_{-1})$ e $P_{L+1}(x_{L+1}, y_{L+1})$. A curva é desenhada apenas de P_0 até P_L, tal como habitualmente. Os pontos fantasma são utilizados somente para o cálculo de $P'(t_k)$. Desta forma, será possível chegar aos valores Sx_0 e Sy_0, para o ponto P_0, e aos valores Sx_L e Sy_L para o ponto P_L (Equação (2.29)).

$$\begin{aligned} Sx_0 &= v(x_1 - x_{-1}) \\ Sy_0 &= v(y_1 - y_{-1}) \\ Sx_L &= v(x_{L+1} - x_{L-1}) \\ Sy_L &= v(y_{L+1} - y_{L-1}) \end{aligned} \tag{2.29}$$

O método Catmull-Rom permite ainda a definição da tensão nos pontos de controlo da curva. Quanto maior a tensão, mais apertada fica a curva ao ponto de controlo. Se a tensão for menor, a curva torna-se mais livre.

Aqui, designaremos a tensão por v_k. Para recorrer a v_k é necessário calcular $P'(t_k)$ de acordo com a seguinte Equação (2.30).

$$P'(t_k) = \frac{1}{2}(1 - v_k)(P_{k+1} - P_{k-1}) \tag{2.30}$$

Quando v_k é igual a zero, v é igual a 0,5. De um modo geral, v_k assume valores entre -1 e 1. No entanto, nada impede de se lhe dar outros valores, tal como se pode ver na Figura 2.54 em que se encontram várias curvas Catmull-Rom, todas com os mesmos pontos de controlo, mas com valores de tensão diferentes. Os pontos de controlo são representados por quadrados.

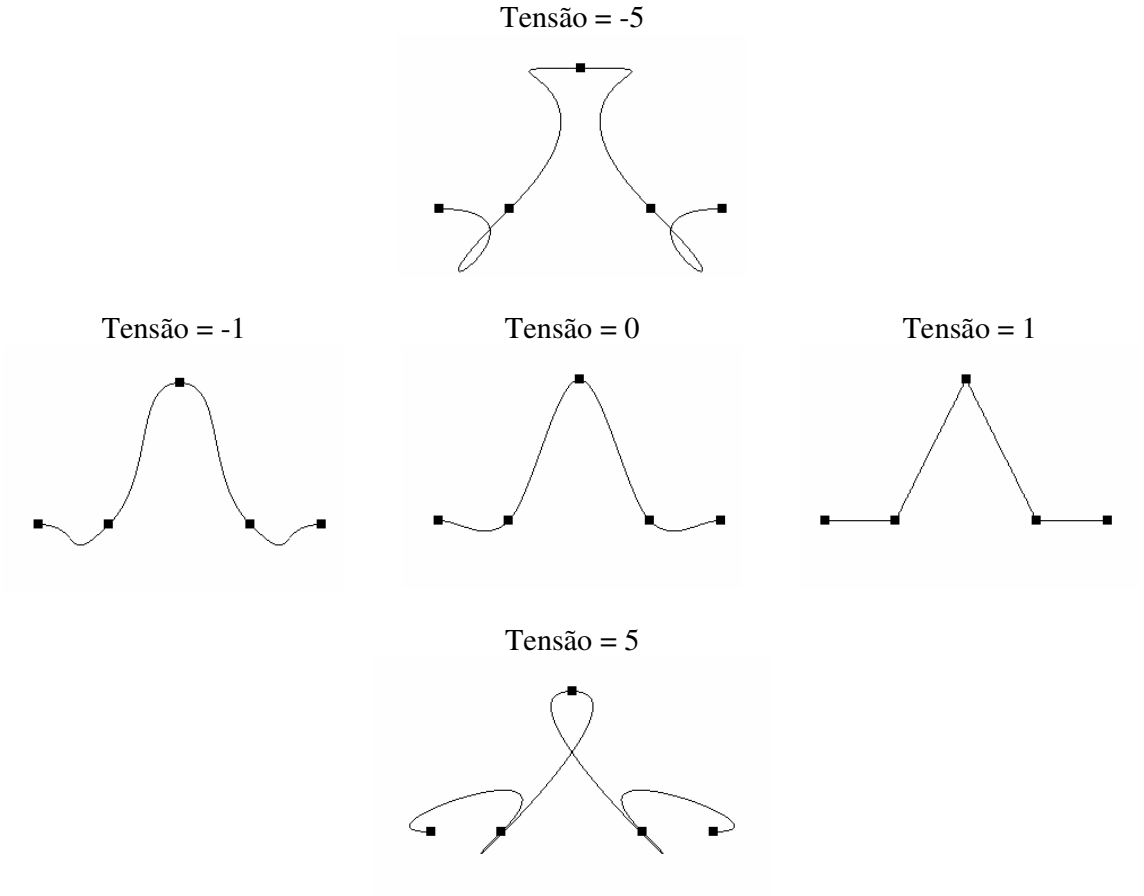


Figura 2.54: Curvas Catmull-Rom com diferentes valores de tensão

Existe ainda uma forma de obter maior controlo sobre a curva, recorrendo ao método Kochanek-Bartels. As curvas Kochanek-Bartels, à semelhança das curvas Catmull-Rom, também são baseadas nas curvas Hermite. Só que, nas curvas Kochanek-Bartels, para além da tensão, dispõe-se de dois parâmetros adicionais: a *bias* e a continuidade. A continuidade controla a continuidade no ponto em que duas secções da curva se encontram, ou seja, num

ponto de controlo. A *bias* é outro parâmetro que afecta o desenho da curva [Fole00, Hear97, Hill01].

Dados quatro pontos de controlo, P_{k-1}, P_k, P_{k+1}, P_{k+2}, as derivadas no ponto P_k e no ponto P_{k+1} podem ser calculadas fazendo uso das Equações indicadas em (2.31) [Fole00, Hear97].

$$\begin{aligned} P'(t_k) &= \frac{1}{2}(1-v_k)\left[(1+b)(1-c)(P_k - P_{k-1}) + (1-b)(1+c)(P_{k+1} - P_k)\right] \\ P'(t_{k+1}) &= \frac{1}{2}(1-v_k)\left[(1+b)(1+c)(P_{k+1} - P_k) + (1-b)(1-c)(P_{k+2} - P_{k+1})\right] \end{aligned} \tag{2.31}$$

v_k é, tal como habitualmente, a tensão.

b é a *bias*.

c representa a continuidade.

Para a determinação das derivadas no primeiro ponto de controlo da curva e no último ponto de controlo da curva pode-se adoptar uma estratégia igual à que já foi anteriormente descrita para as curvas Catmull-Rom.

Na Figura 2.55 pode-se observar o efeito da *bias* no desenho de uma curva.

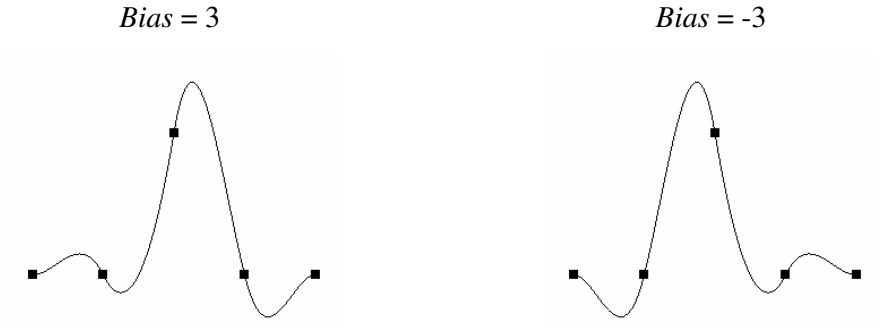


Figura 2.55: Curvas com diferentes valores de *Bias*

As curvas Kochanek-Bartels foram especialmente desenhadas para a geração de caminhos, pelo que são um forte candidato para a construção do caminho de baixo nível dentro de cada célula do modelo [Fole00, Hear97].

2.3.4.4. Escolha da curva mais conveniente para o caminho de baixo nível

Agora, que já se apresentaram as várias curvas através das quais se poderá gerar o caminho de baixo nível dentro de cada célula, há que escolher uma delas.

A curva Bézier, como já foi referido anteriormente, é um tipo particular de B-Spline, pelo que a escolha é entre a curva B-Spline e as curvas Hermite.

Entre as curvas Hermite, a curva Kochaneck-Bartels é aquela onde se consegue maior controlo sobre o desenho da curva. Isto porque nesta curva, dispõe-se dos parâmetros tensão, *bias* e continuidade. As curvas Catmull-Rom, por seu turno, só permitem controlar a tensão. Além disso, as curvas Kochanek-Bartels foram especialmente desenvolvidas para a geração de caminhos no contexto da animação. Deste modo, entre as curvas Hermite, parece-nos que a Kochaneck-Bartels é a melhor opção. No entanto, será a Kochaneck-Bartels melhor ou pior do que a B-Spline para efeitos do desenho do caminho de baixo nível?

Recordamos que a curva B-Spline é uma curva que aproxima os pontos. A curva Kochaneck-Bartels passa obrigatoriamente por todos os pontos. Ora, o facto de a Kochaneck-Bartels interpolar todos os pontos poderá criar algumas situações indesejáveis, caso exista uma grande abundância e densidade de pontos, o que é o caso do caminho de baixo nível. Na Figura 2.56 podemos observar algumas situações em que acontece este problema.

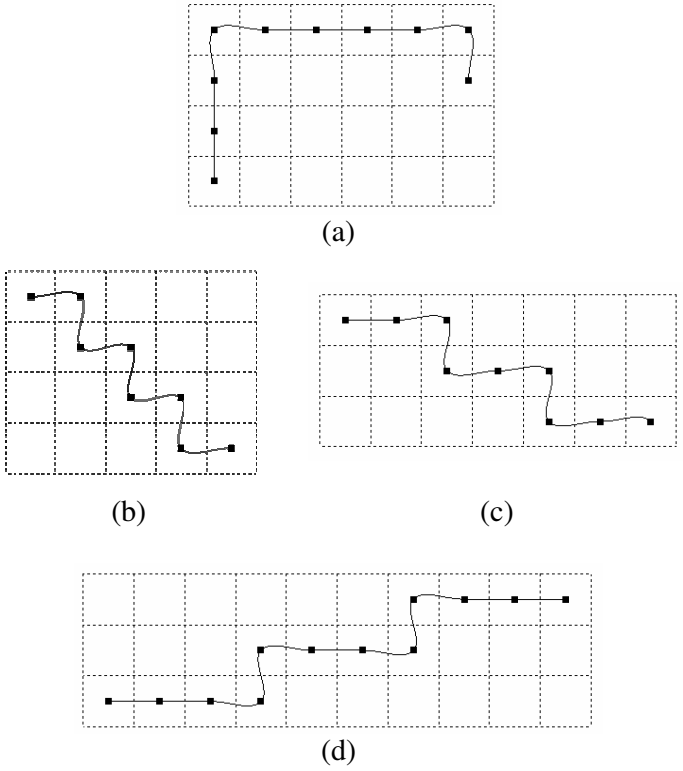


Figura 2.56: Caminhos de baixo nível desenhados com curvas Kochaneck-Bartels

Na Figura 2.56 encontram-se quatro caminhos de baixo nível, todos gerados com recurso às curvas Kochanek-Bartels. Recordamos que os pontos de controlo, representados por quadrados no caminho de baixo nível, são os centros dos *voxels* através dos quais esse caminho passa. O corte horizontal da grelha de *voxels* é representado a traço interrompido. Neste momento, o que nos interessa avaliar é a forma da curva do caminho. Por isso, não assinalámos qual é o primeiro ponto e o último ponto do caminho porque isso, para este caso, é irrelevante, e iria apenas acrescentar complexidade à Figura.

No caminho de baixo nível (a), da Figura 2.56, a disposição dos pontos de controlo não cria uma situação indesejável. O utilizador segue sempre a direito, à excepção dos dois pontos intermédios onde muda de direcção.

No caminho de baixo nível (b), da Figura 2.56, a situação já não é tão satisfatória. Repare-se que, devido à disposição dos pontos de controlo, e uma vez que é necessário passar por todos eles, o utilizador é forçado a curvar constantemente para a esquerda e para direita. Não será um percurso nada agradável.

Nos caminhos de baixo nível (c) e (d) da Figura 2.56 repetem-se situações semelhantes. O utilizador anda aos "esses" resultando daí uma sensação de desconforto.

No entanto, se em vez de recorremos às curvas Kochaneck-Bartels, recorrermos às curvas B-Spline, o caminho torna-se menos curvilíneo. Na

Figura 2.57 encontram-se quatro caminhos de baixo nível, gerados com os mesmos pontos de controlo dos da Figura 2.56.

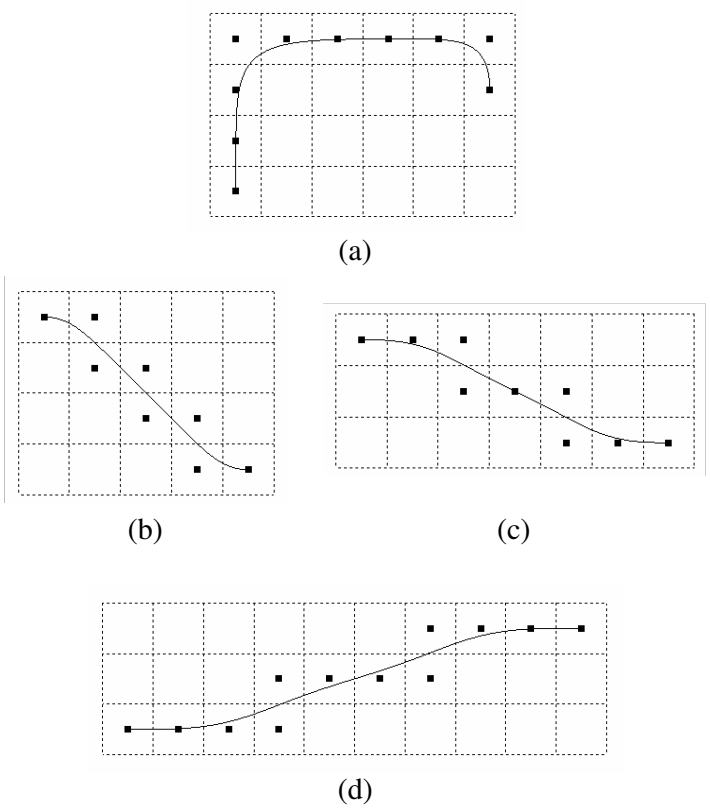


Figura 2.57: Caminhos de baixo nível desenhados com curvas B-Spline

Tal como se pode depreender pela Figura 2.56 e pela Figura 2.57, os caminhos desenhados com as curvas B-Spline reduzem significativamente os desvios para a esquerda e para a direita em relação aos caminhos desenhados com as curvas Kochaneck-Bartels.

C. Andújar gerou o caminho de baixo nível recorrendo a uma interpolação de Hermite, escolhendo um ponto de controlo a cada dois pontos do caminho [Andú04]. Descartar um ponto de controlo a cada dois pontos resolve o caso (b) da Figura 2.56. No entanto, nos casos (c) e (d) da Figura 2.56 este modo de proceder não é satisfatório, continuando o utilizador a ser obrigado a curvar desnecessariamente para a esquerda e para a direita.

Por isso, embora a curva Kochaneck-Bartels tenha sido especificamente desenvolvida para a geração de caminhos no contexto da animação, no caso particular do caminho de baixo nível, parece-nos que a melhor opção é a curva B-Spline.

2.3.5. Redução dos desvios desnecessários no caminho de baixo nível

Na secção anterior demonstrou-se que a melhor curva para a geração do caminho de baixo nível é a B-Spline. No entanto, esta curva não é, ainda assim, capaz de resolver inteiramente o problema relativo aos sucessivos desvios do utilizador para a esquerda e para a direita. Repare-se na Figura 2.57 (d), em que o parâmetro *m* da curva B-Spline é igual a 12. Se se olhar com atenção, nota-se que ainda existem ligeiros desvios para a esquerda e para a direita. Se *m* diminuir, de 12 para 8, esses desvios acentuam-se, tal como se pode observar na Figura 2.58. O corte horizontal da grelha de *voxels* é representado a traço interrompido.

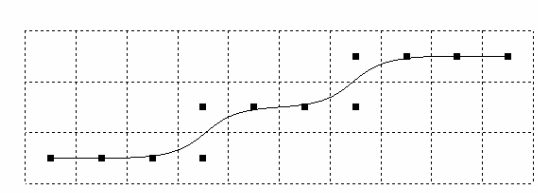


Figura 2.58: Caminho de baixo nível desenhado com curva B-Spline e com o parâmetro *m* igual a oito

Uma vez que a curva B-Spline, por si só, não resolve o problema dos desvios desnecessários, é preciso recorrer a medidas adicionais. Ora, há que reparar que a disposição dos pontos de controlo, na Figura 2.58, segue um padrão constante. A cada cinco pontos, o padrão repete-se. Na Figura 2.59 (a) estão assinaladas as zonas de repetição dos padrões. A zona da terceira repetição, assinalada com pontos azuis, é incompleta, uma vez que o quinto ponto não se repete.

Dado que existe este padrão, pode estabelecer-se que os pontos da Figura 2.58 configuram um possível segmento de recta. Esse segmento de recta começaria no primeiro ponto de controlo e acabaria no último ponto de controlo (Figura 2.59 (b)). Se o utilizador seguisse ao longo deste hipotético segmento de recta, os desvios desnecessários no caminho seriam totalmente eliminados.

Portanto, se houvesse uma maneira de reconhecer quando os pontos de controlo configuram um segmento de recta, isso seria vantajoso. No entanto, não é simples reconhecer este tipo de situações. Por exemplo, na Figura 2.60, não existe um padrão na disposição dos pontos de controlo que se mantenha desde o início da curva até ao seu fim. Se, erradamente, uníssemos o primeiro ponto de controlo ao último ponto de controlo através de um segmento de recta, de forma a eliminar todos os desvios, isso faria o utilizador chocar com a parede, representada a cinzento. Ora, pretende-se eliminar apenas os desvios desnecessários. No caso da Figura 2.60 há desvios que são indispensáveis para evitar atravessar obstáculos e, esses, não podem ser eliminados.

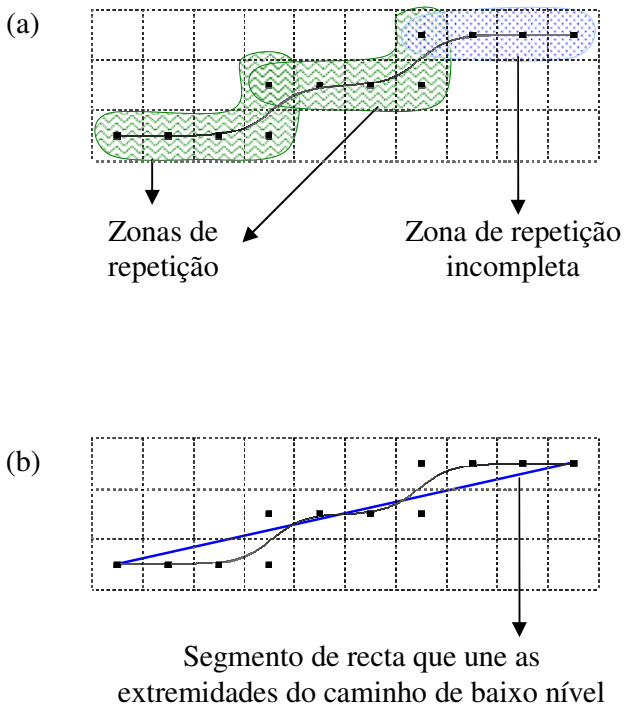


Figura 2.59: Caminho de baixo nível com as zonas de repetição assinaladas

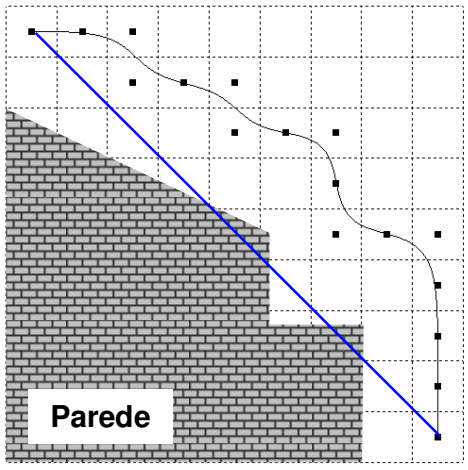


Figura 2.60: Segmento de recta incorrecto no caminho de baixo nível

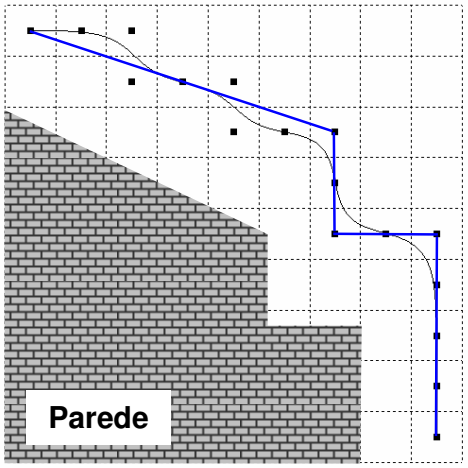


Figura 2.61: Segmentos de recta para caminho de baixo nível

No entanto, embora na sua totalidade os pontos de controlo da Figura 2.60 não configurem um segmento de recta, existem subconjuntos destes pontos para os quais isso pode acontecer, tal como se mostra na Figura 2.61. Cada segmento de recta corresponde ao conjunto de pontos de controlo intermédios situados entre as suas duas extremidades. Na Figura 2.61 o caminho de baixo nível foi decomposto em quatro segmentos de recta assinalados a azul. Em cada segmento de recta, existe um padrão na disposição dos pontos de controlo.

Sendo assim, podia estabelecer-se que, sempre que se encontrasse um padrão de posicionamento dos pontos repetitivo, estar-se-ia perante uma linha recta. Isso resultaria, tanto na Figura 2.58 como na Figura 2.61, em que existem padrões de posicionamento repetitivos simples de detectar. Só que nem sempre estes padrões repetitivos são fáceis de se reconhecer. Na Figura 2.62 encontra-se um conjunto de pontos que, no seu conjunto, pode dar origem a uma linha. No entanto, seria bastante difícil detectar o padrão repetitivo de posicionamento neste conjunto de pontos em particular.

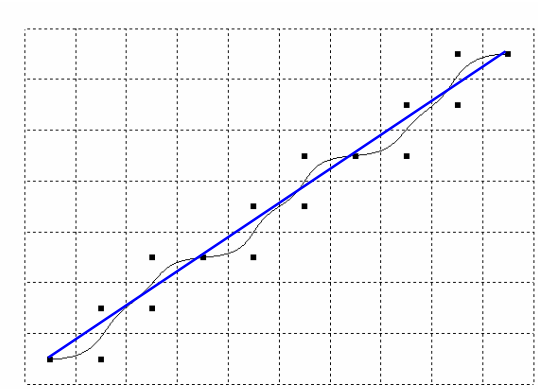


Figura 2.62: Caminho de baixo nível em que o padrão repetitivo de posicionamento dos pontos de controlo é difícil de detectar

O problema que se põe aqui é o inverso daquele que o algoritmo de Bresenham resolve. No algoritmo de Bresenham, a partir dos pontos inicial e final de um segmento de recta, tem de se decidir quais os pixels a colorir no ecrã, de maneira a que esses pixels, no seu conjunto, desenhem uma linha [Fole00, Hill01]. No nosso caso, podemos fazer a equivalência entre os *voxels* e os pixels. Parte-se de um conjunto de *voxels*, que servem de base ao caminho de baixo nível, e pretende-se descobrir se alguns desses *voxels* configuram possíveis segmentos de recta.

Para a resolução deste problema concebemos um algoritmo ao qual demos o nome de "Melhoramento de Caminhos". Este algoritmo recebe um conjunto de pontos ordenados, que correspondem aos *voxels*, e consegue reconhecer quando um determinado subconjunto desses pontos dão origem a um segmento de recta. O conjunto de pontos ordenados é referenciado no algoritmo como *Caminho*, uma vez que é com base nestes pontos que se desenha o caminho de baixo nível. O número de pontos deve ser superior a 1. O algoritmo de Melhoramento de Caminhos desenvolve-se ao longo dos seguintes passos:

1. Começar no primeiro ponto do *Caminho*. O primeiro ponto do *Caminho* é o ponto P_0.
2. Criar um apontador, *i*, para o primeiro ponto do caminho, ou seja, P_0. O ponto para o qual *i* aponta é P_i.
3. Se P_i for o último ponto do caminho, então, avançar para o passo 7.
4. Avançar o apontador *i* para o próximo ponto do *Caminho*.
5. Verificar se os pontos intermédios entre P_0 e o P_i formam um segmento de recta. Para isso, procede-se da seguinte maneira:
   - 5.1. Unir P_0 a P_i através de um segmento de recta.
   - 5.2. Se todos os *voxels* de P_{0+1} até P_{i-1} forem intersectados pelo segmento de recta que une P_0 a P_i, então, o conjunto de pontos de P_0 a P_i configura um segmento de recta.
     - 5.2.1. Por agora, ainda não se faz mais nada, porque este segmento ainda pode, possivelmente, abranger mais pontos.
   - 5.3. Se nem todos os *voxels* de P_{0+1} até P_{i-1} forem intersectados pelo segmento de recta que une P_0 a P_i, então o conjunto de pontos de P_0 a P_i não configura um segmento de recta. No entanto, o conjunto de pontos de P_0 a P_{i-1} configura um segmento de recta.
     - 5.3.1. Descobre-se que existe um segmento de recta de P_0 a P_{i-1}.
     - 5.3.2. P_{i-1} passa a funcionar, para este algoritmo, como um novo P_0.
     - 5.3.3. Tenta-se descobrir mais um segmento de recta a partir de P_{i-1}.
6. Caso P_i não seja o último ponto do *Caminho*, voltar ao passo 4.
7. Os pontos que vão de P_0 até ao ponto P_i, configuram o último segmento de recta do caminho.

Vamos agora clarificar um pouco mais o passo 5.2 do algoritmo de Melhoramento de Caminhos.

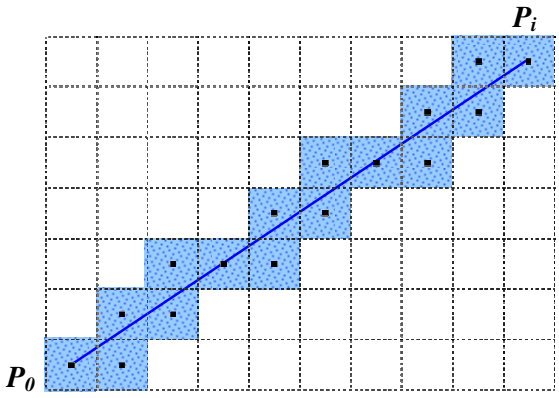
Tal como se refere neste passo, para que os pontos de P_0 a P_i configurem um segmento de recta é necessário que todos os *voxels* de P_{0+1} até P_{i-1} sejam intersectados pelo segmento de recta que une P_0 a P_i. Esta situação é mais fácil de visualizar em duas dimensões.

Repare-se na Figura 2.63 (a). Os *voxels* correspondentes aos pontos entre P_{0+1} e P_{i-1} são todos intersectados pelo segmento de recta que une P_0 a P_i. Portanto, os pontos de P_0 a P_i formam, no seu conjunto, um segmento de recta.

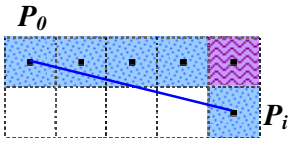
Na Figura 2.63 (b) nem todos os *voxels* correspondentes aos pontos entre P_{0+1} e P_{i-1} são intersectados, pelo que não se está perante um segmento de recta.

A situação da Figura 2.64 é especial. O padrão repetitivo de posicionamento dos pontos denuncia um segmento de recta. No entanto, nem todos os *voxels* correspondentes aos pontos entre P_{0+1} e P_{i-1} são intersectados pelo segmento de recta que une P_0 a P_i. Para resolver esta situação, estabelece-se um valor de tolerância que aumenta o volume do *voxel*. Através de sucessivas experiências, o valor de tolerância que nos forneceu melhores resultados foi o de 20%. Sendo assim, a largura, a altura e a profundidade dos *voxels* são aumentadas em 20%. Desta forma, todos os *voxels* correspondentes aos pontos entre P_{0+1} e P_{i-1} passam a ser intersectados.

Na Figura 2.65 os pontos de P_0 a P_i configurem um segmento de recta. Repare-se que, na situação exemplificada nesta Figura também é permitido à câmara mover-se na diagonal.



(a)



(b)

Voxels intersectados pelo segmento de recta que une P_0 a Pi

Voxel não intersectado pelo segmento de recta que une P_0 a Pi

Figura 2.63: Passo 5.2. do algoritmo de Melhoramento de Caminhos

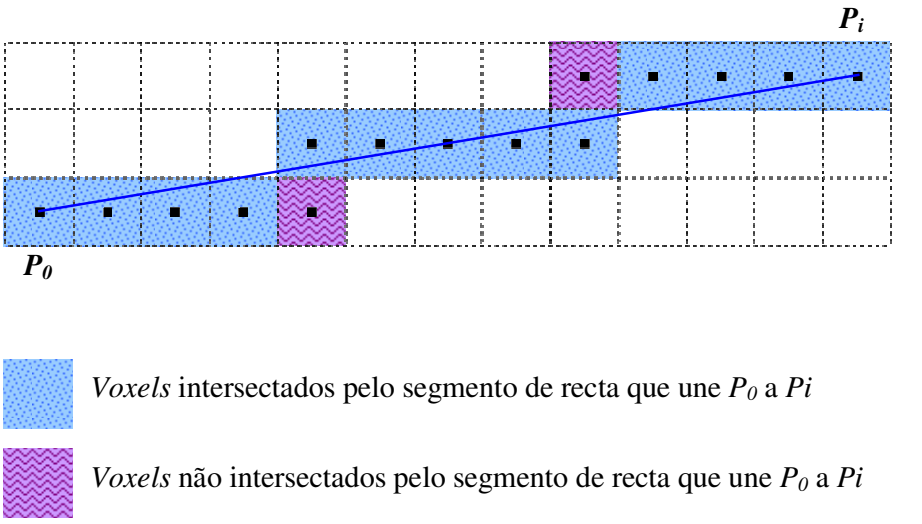


Figura 2.64: Situação especial do passo 5.2. do algoritmo de Melhoramento de Caminhos

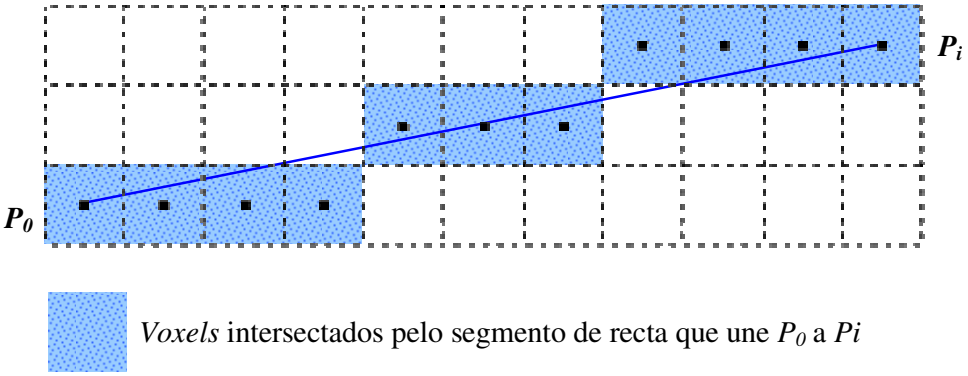


Figura 2.65: Passo 5.2. do algoritmo de Melhoramento de Caminhos, onde a câmara, além de se mover na horizontal e na vertical, também se pode mover na diagonal

Na Figura 2.66 encontra-se a aplicação do algoritmo de Melhoramento de Caminhos a um conjunto de pontos. Através de uma primeira aplicação do algoritmo descobre-se um primeiro segmento de P_0 a P_{15}. Depois de se estabelecer que P_{15} é o novo ponto inicial, pode-se descobrir outro novo segmento de P_{15} a P_{18}. De P_{18} a P_{23} existe um novo segmento. De P_{23} a P_{27} existe outro. De P_{27} a P_{29} encontra-se outro. E, finalmente, de P_{29} a P_{30} surge um último segmento.

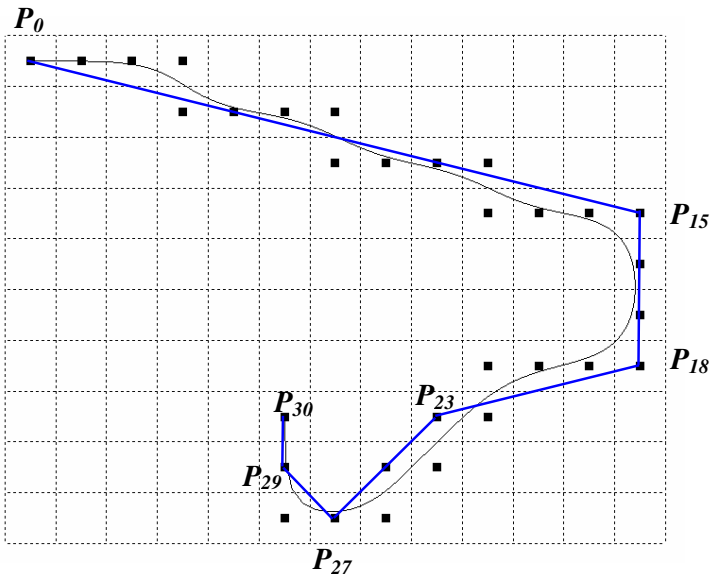


Figura 2.66: Algoritmo de Melhoramento de Caminhos aplicado a um conjunto de pontos

A aplicação do algoritmo de Melhoramento de Caminhos ao conjunto de pontos da Figura 2.60 produz o resultado já apresentado na Figura 2.61.

Há agora que estabelecer como incorporar estes segmentos de recta no caminho de baixo nível. Não basta, simplesmente, deslocar o utilizador ao longo dos segmentos de recta, uma vez que chegado ao término de um segmento efectuar-se-ia uma desagradável e abrupta mudança de direcção ao transitar para o próximo segmento. O que sugerimos é, primeiro que tudo, para cada um dos segmentos, anular os pontos de controlo que lhe deram origem. Esses pontos serão substituídos por outros pontos, posicionados de forma uniforme ao longo do segmento de recta. O número de pontos a posicionar ao longo do segmento é o seguinte:

$$N = \frac{D_{P_0 P_{Fim}}}{L} + 1 \tag{2.32}$$

N é o nº de pontos de controlo a posicionar ao longo do segmento de recta.

$D_{P_0 P_{Fim}}$ é a distância entre o primeiro ponto do segmento e o último ponto do segmento.

L é o comprimento da aresta do *voxel* que, como se sabe, é um cubo.

N é arredondado para zero casas decimais, de modo a que o número de pontos seja sempre um inteiro.

Para calcular $D_{P_0 P_{Fim}}$ recorre-se à Equação (2.33). Esta Equação calcula a distância entre o ponto $A = (x_A, y_A)$ e o ponto $B = (x_B, y_B)$.

$$D = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2} \tag{2.33}$$

Em três dimensões, a distância entre os pontos $A=(x_A, y_A, z_A)$ e $B=(x_B, y_B, z_B)$, passa a ser:

$$D=\sqrt{(x_B-x_A)^2+(y_B-y_A)^2+(z_B-z_A)^2} \tag{2.34}$$

Após sabermos quantos pontos de controlo é necessário posicionar entre o início do segmento de recta e o fim do segmento de recta, há que, efectivamente, posicioná-los. Para tal, pode recorrer-se a uma interpolação linear entre o ponto inicial do segmento (P_0) e o ponto final do segmento (P_{Fim}). Recordamos que a interpolação linear é efectuada recorrendo à Equação (2.18). Para obter os pontos de controlo entre P_0 e P_{Fim} faz-se variar o parâmetro t entre 0 e 1. O incremento no valor de t, até se chegar a 1, é dado pela Equação (2.35).

$$I=\frac{1}{N-1} \tag{2.35}$$

I é o incremento entre um valor de t e o seguinte. A cada incremento I em t posiciona-se um novo ponto de controlo entre P_0 e P_{Fim}.

N é o número de pontos de controlo a posicionar ao longo do segmento de recta.

N é obtido recorrendo à Equação (2.32).

A distância entre um ponto de controlo e o seguinte, no caminho original, é sempre igual ao comprimento da aresta do *voxel*, caso a câmara se possa deslocar apenas na horizontal e na vertical. Tenta-se que a distância entre um ponto de controlo e o seguinte, ao longo do segmento de recta, seja, também, o mais próxima possível do comprimento da aresta do *voxel*. Por essa razão se divide $D_{P_0P_{Fim}}$ por L na Equação (2.32).

Quando é permitido à câmara mover-se também na diagonal, então aí a distancia entre um ponto de controlo e o seguinte poderá ser igual a L ou igual a $L\sqrt{2}$. Ainda assim, na Equação (2.32), continua a dividir-se $D_{P_0P_{Fim}}$ por L, porque considerámos que L é a medida padrão para a colocação dos pontos de controlo ao longo dos segmentos de recta.

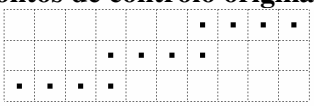
Há que realçar que o número de pontos de controlo modificados (N) pode ser diferente do número de pontos de controlo originais. Isto acontece para que a distância entre um ponto de controlo e o seguinte, ao longo do segmento de recta, seja o mais próxima possível de L. Normalmente, não se consegue que o espaçamento entre os pontos, ao longo do segmento de recta, seja igual à aresta do *voxel*, mas apenas aproximadamente igual.

No Quadro 2.7 apresentam-se os pontos de controlo de caminhos simples, em que existe apenas um único segmento de recta, e os pontos de controlo modificados para esses caminhos simples.

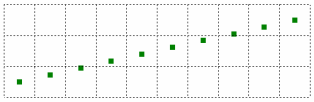
Repare-se que só no último caso, do Quadro 2.7, é que a distância entre os pontos de controlo modificados é igual à aresta do *voxel*. No entanto, neste caso particular, a modificação dos pontos de controlo é desnecessária, uma vez que todos pontos de controlo originais já pertencem ao segmento de recta. Por essa razão, os pontos de controlo originais e os pontos de controlo modificados são os mesmos. Quanto a $D_{P_0P_{Fim}}$, N, L e I, eles são os parâmetros necessários à aplicação das Equações (2.32) e (2.35).

Modificação de pontos de controlo em caminhos

Pontos de controlo originais



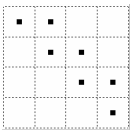
Pontos de controlo modificados



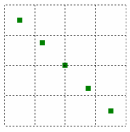
$D_{P_0P_{Fim}} \approx 9.219544$ $N = 10$ $L = 1$ $I = \frac{1}{10-1} \approx 0{,}111111$

Distância entre um ponto de controlo e o seguinte nos pontos de controlo modificados $\approx 1{,}024394$

Pontos de controlo originais



Pontos de controlo modificados

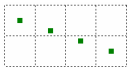


$D_{P_0P_{Fim}} \approx 4{,}242641$ $N = 5$ $L = 1$ $I = \frac{1}{5-1} = 0{,}25$

Distância entre um ponto de controlo e o seguinte nos pontos de controlo modificados $\approx 1{,}060660$

Pontos de controlo originais

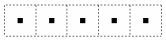
Pontos de controlo modificados



$D_{P_0P_{Fim}} \approx 3.162278$ $N = 4$ $L = 1$ $I = \frac{1}{4-1} = 0{,}333333$

Distância entre um ponto de controlo e o seguinte nos pontos de controlo modificados $\approx 1{,}054093$

Pontos de controlo originais



Pontos de controlo modificados

$D_{P_0P_{Fim}} = 4$ $N = 5$ $L = 1$ $I = \frac{1}{5-1} = 0{,}25$

Distância entre um ponto de controlo e o seguinte nos pontos de controlo modificados = 1

Quadro 2.7: Modificação de pontos de controlo em caminhos

No Quadro 2.7 os pontos de controlo apresentados pertencem a caminhos simples em que existe apenas um único segmento de recta. No entanto, em caminhos mais complexos existem vários segmentos de recta. É o caso da Figura 2.61, em que os pontos de controlo dão origem a 4 segmentos de recta. Para cada um dos segmentos de recta, é necessário efectuar a necessária transformação dos pontos de controlo. O resultado é o que se pode observar na Figura 2.67. Repare-se que, na realidade, os pontos de controlo só são modificados no primeiro segmento de recta, logo ao cimo da Figura. Nos restantes segmentos de recta, os pontos de controlo originais coincidem com os pontos de controlo modificados. Isto acontece porque, nos restantes três segmentos os pontos de controlo originais já pertencem ao segmento de recta ao qual irão dar origem.

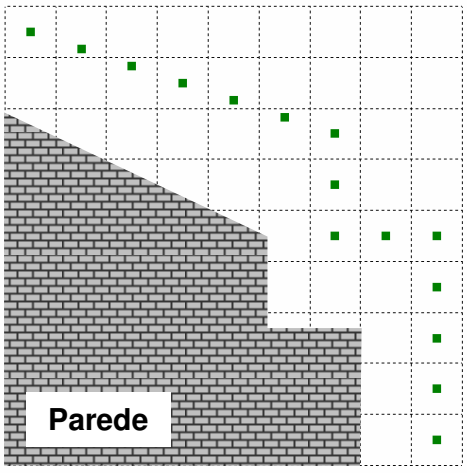


Figura 2.67: Pontos de controlo modificados do caminho da Figura 2.61

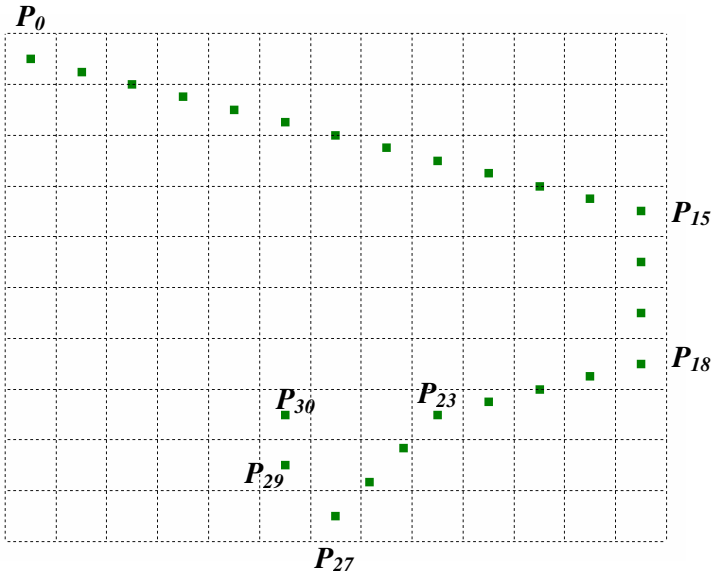


Figura 2.68: Pontos de controlo modificados do caminho da Figura 2.66

No caminho da Figura 2.66 existem também vários segmentos de recta. Após modificação dos pontos de controlo, obtém-se o resultado da Figura 2.68. Aqui, existe modificação dos pontos de controlo nos seguintes segmentos de recta: segmento de recta de P_0 a P_{15}, segmento de recta de P_{18} a P_{23}, segmento de recta de P_{23} a P_{27} e segmento de recta de P_{27} a P_{29}. No segmento de recta P_{15} a P_{18} e no segmento de recta P_{29} a P_{30} os pontos de controlo originais coincidem com os pontos de controlo modificados porque os originais já pertenciam ao segmento de recta.

Depois de se efectuar a modificação dos pontos de controlo, desenha-se a curva para o caminho de baixo nível com base no novo conjunto de pontos. Na Figura 2.69 encontra-se desenhado o caminho de baixo nível com base nos pontos de controlo modificados da Figura 2.67. Como se pode observar, em relação à Figura 2.61, os desvios desnecessários, ao longo dos pontos de controlo correspondentes ao segmento de recta ao cima da Figura, foram eliminados. Conservam-se, no entanto, os desvios que evitam que o utilizador choque com os obstáculos arquitectónicos.

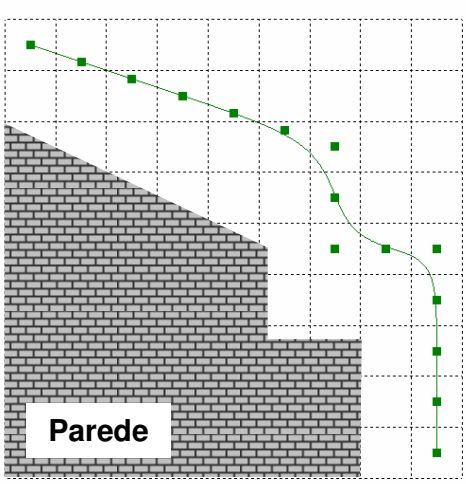


Figura 2.69: Caminho de baixo nível desenhado a partir dos pontos de controlo modificados da Figura 2.67

Na Figura 2.70 encontra-se o caminho de baixo nível desenhado com base nos pontos de controlo modificados da Figura 2.68. Como se pode observar, os desvios desnecessários também foram eliminados.

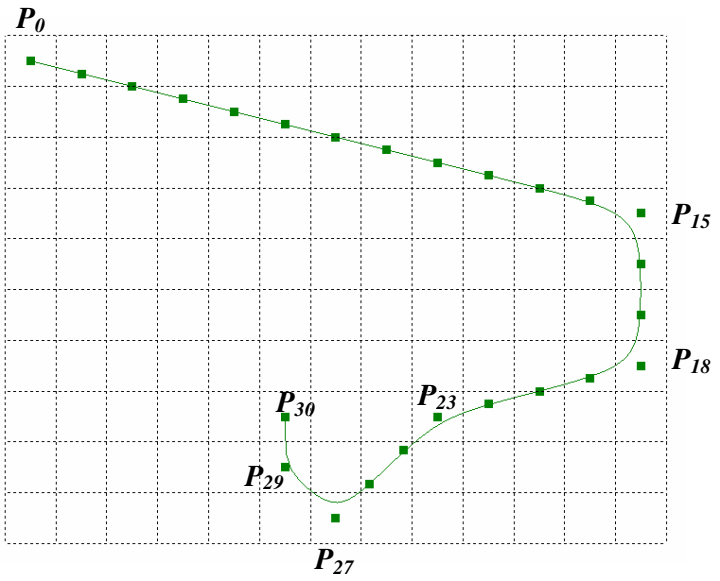


Figura 2.70: Caminho de baixo nível desenhado a partir dos pontos de controlo modificados da Figura 2.68

Claro, se o caminho se reduzir a um único segmento de recta, na realidade não existirá uma curva, mas simplesmente um segmento que une o primeiro ponto de controlo ao último ponto de controlo. Esse é o caso de todos os caminhos apresentados no Quadro 2.7.

Portanto, revendo, a redução dos desvios necessários num caminho processa-se ao longo das seguintes fases:

1. Detectar quais os pontos de controlo que podem configurar possíveis segmentos de recta no caminho. Para este efeito, recorre-se ao algoritmo de Melhoramento de Caminhos.
2. Para cada segmento de recta encontrado no caminho, proceder à alteração dos pontos de controlo. Para tal, recorre-se à Equação (2.32), para saber quantos pontos de controlo posicionar entre o início e o fim do segmento de recta. Após se saber quantos pontos de controlo posicionar, utiliza-se uma interpolação linear para distribuir esses pontos de controlo. A distância entre um ponto de controlo e o seguinte é o mais próxima possível do comprimento da aresta do *voxel*.
3. Finalmente, desenha-se o novo caminho tendo por base os pontos de controlo modificados. Neste novo caminho foram eliminados os desvios desnecessários para a esquerda e para a direita, conservando-se apenas os essenciais para evitar chocar com obstáculos do modelo.

Desenhado o caminho de baixo nível, o utilizador desloca-se através dele. Para fazer o utilizador deslocar-se ao longo da curva do caminho, varia-se o parâmetro t da Equação (2.22), a Equação da curva B-Spline. A diferença entre um valor t e o seguinte é sempre a mesma. Quanto menor for essa diferença, mais fluida é a sensação do movimento.

Na Figura 2.70 (a) encontram-se os pontos de controlo originais de um caminho e, assinalados a azul, os segmentos de recta que esses pontos configuram. Na Figura 2.70 (b) estão os pontos de controlo modificados. Na Figura 2.70 (c) está desenhada a curva que tem por base os

pontos de controlo modificados. Assinaladas, com losangos a roxo, estão as posições do utilizador, obtidas através da variação do parâmetro *t* da Equação (2.22).

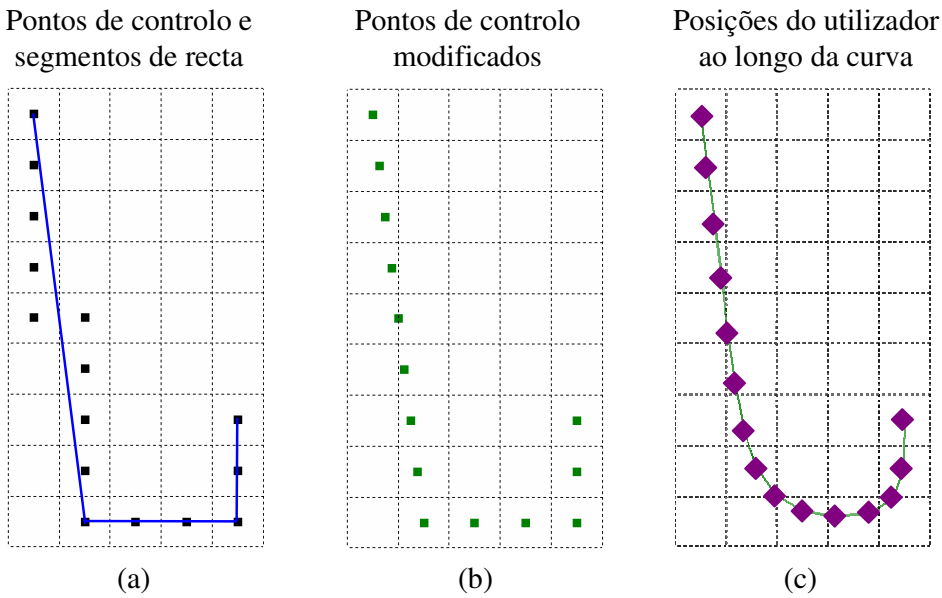


Figura 2.71: Posições do utilizador ao longo de um caminho

Depois, há que saber qual a orientação da câmara em cada uma das posições do utilizador ao longo da curva. Esta questão irá ser abordada seguidamente.

2.3.6. Orientação da câmara ao longo do caminho de baixo nível

Depois de se terem gerado as várias posições da câmara ao longo do caminho de baixo nível, há que decidir a orientação da câmara em cada uma dessas posições. A orientação da câmara em cada posição deve ser aquela que mostre maior quantidade de informação ao utilizador.

Para determinar a orientação da câmara nas várias posições do caminho de baixo nível de uma célula, de acordo com [Andú04], procede-se da seguinte maneira:

1. Colocar a câmara no ponto onde se inicia o caminho de baixo nível da célula, apontando-a para o próximo ponto do caminho.

   Recordamos que este ponto é indicado pelo utilizador, caso se esteja na primeira célula do caminho de alto nível. Nas restantes células, o primeiro ponto do caminho de baixo nível pertence a um portal.

2. Avançar a câmara para a próxima posição do caminho de baixo nível.

3. Avaliar a melhor orientação da câmara nesta nova posição.

   Na nova posição, a câmara poderá:

   - Manter a mesma orientação que na posição anterior.
   - Mudar de orientação, rodando para a esquerda.

- Mudar de orientação, rodando para a direita.
- Mudar de orientação, rodando para cima.
- Mudar de orientação, rodando para baixo.

Estas possíveis orientações da câmara podem ser visualizadas na Figura 2.72.

A rotação da câmara, para a esquerda, para a direita, para cima ou para baixo deve ser pequena. Se se aplicar uma rotação de muitos graus, o movimento não será suave, o que é desagradável para o utilizador.

Entre as cinco possibilidades mencionadas, há que escolher a melhor. Para isso, avalia-se a quantidade de informação visível através de cada uma das orientações. Podia-se determinar que a orientação em que a qualidade da vista é melhor seria a escolhida. No entanto, isto pode ser pernicioso, pois, caso a célula tenha uma zona de elevado interesse, a câmara tenderá a estar sempre apontada para essa zona. Sendo assim, na escolha da orientação da câmara, há que ter em conta, não apenas a quantidade de informação visível, mas também o historial das faces já anteriormente vistas.

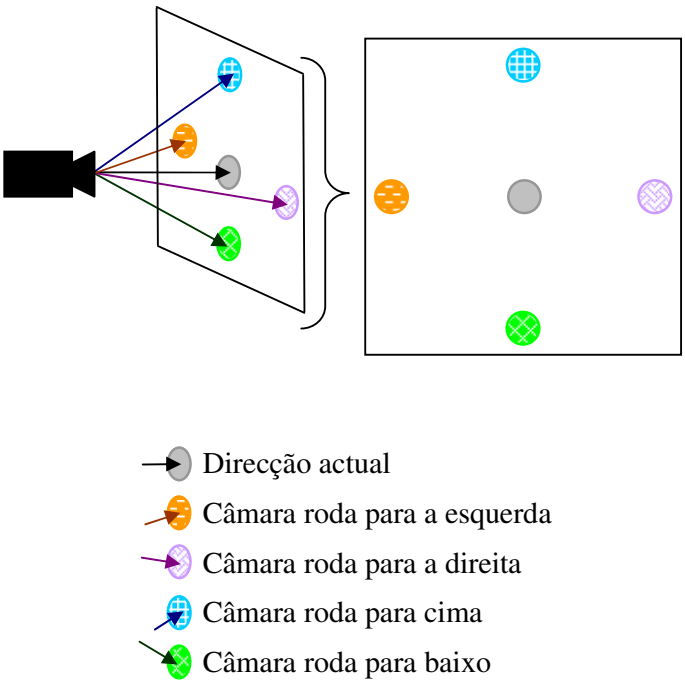


Figura 2.72: Possíveis mudanças de orientação da câmara ao longo do caminho de baixo nível

Adaptado de: C. Andújar, P. Vázquez e M. Fairén. Way-Finder: guided tours through complex walkthrough models. Em Eurographics, Volume 23, Number 3, 2004.

C. Andújar refere, em [Andú04], que uma face é considerada como já vista após ter sido visualizada por, pelo menos, vinte vezes. Após essas vinte vezes, para efeitos do cálculo da qualidade da vista, essa face passa a ser considerada como parte do fundo. Assim, evita-se que a câmara fique viciada numa determinada região da célula.

É aconselhável considerar uma face como vista apenas após vinte visualizações. Se uma face é marcada como vista após a primeira visualização, isso poderá fazer com que a câmara esteja constantemente a mudar de direcção, o que também é incómodo para o utilizador. Um limite de vinte vezes permitirá ao utilizador olhar com atenção para uma determinada região, antes da câmara começar a preferir outras regiões dentro da célula.

Após isto, roda-se a câmara para a orientação escolhida e, caso ainda não se tenha chegado ao fim do caminho de baixo nível, volta-se ao passo 2, avançando a câmara para a próxima posição.

No que se refere ao procedimento apresentado, há ainda que ter em conta as ocasiões em que a câmara chega a um dos pontos de interesse da célula. Caso a posição onde a câmara esteja situada seja um dos pontos interessantes da célula, então a câmara imobiliza-se e gira em torno de si própria, permitindo ao utilizador observar a célula desta posição que foi escolhida como privilegiada. Após esta rotação estar completa, avança-se para a próxima posição no caminho de baixo nível [Andú04].

Convém também estar alerta para o facto de certas orientações da câmara poderem ser consideradas como pouco naturais pelo utilizador, causando-lhe estranheza. Por exemplo, se a câmara estiver virada para trás, enquanto se avança para a frente, o utilizador pode sentir-se pouco confortável. Por isso, a câmara não deverá afastar-se mais do que 30 a 40 graus da direcção ao longo da qual o utilizador está a avançar [Andú04].

Além disso, de uma forma geral, quando atravessam uma porta, as pessoas tendem a olhar em frente e não para os lados. De forma a transmitir uma sensação de deslocamento natural, a câmara deverá também, ao atravessar um portal, estar apontada para a frente, na mesma direcção ao longo da qual se está a avançar. Para atingir este objectivo, à medida que a câmara se aproxima de um portal, a sua rotação é limitada, forçando a que esta passe a apontar em frente [Andú04].

Após se ter determinado a orientação da câmara para cada uma das posições de todos os caminhos de baixo nível, das várias células, o caminho está completo. O utilizador poderá, agora, ser conduzido ao longo do modelo numa visita guiada.

3. Visita guiada a um museu

No âmbito desta dissertação foi desenvolvido um protótipo que, através da implementação dos algoritmos estudados, permite efectuar uma visita guiada a um museu de pintura.

O programa da visita guiada é constituído pelos seguintes módulos:

- Geometria
  - Contém uma série de funções geométricas úteis para lidar com vectores, pontos, polígonos, rectas, curvas, entre outros elementos geométricos.
- Imagem
  - Permite trabalhar com imagens que sejam mapas de bits. As funções contidas neste módulo são especialmente úteis para lidar com os ficheiros de imagem correspondentes às pinturas que estão expostas no museu virtual.
- Edifício
  - Lê, do disco, ou de outro suporte de armazenamento, o modelo através do qual irá ser efectuada a visita guiada.
  - Este módulo recorre ao módulo imagem para carregar as texturas correspondentes às pinturas que estão expostas no museu virtual.
- Algoritmos de pesquisa
  - Contém os algoritmos de pesquisa necessários para a geração do caminho de alto nível e do caminho de baixo nível. Recordamos que o caminho de alto nível é a sequência de células que o utilizador irá visitar. Para cada célula existe um caminho de baixo nível. O caminho de baixo nível é o percurso do utilizador dentro da célula. Na visita guiada ao museu, este percurso leva o utilizador até ao melhor ponto de vista da célula e, seguidamente, permite-lhe observar as várias pinturas expostas nessa mesma célula.
- Qualidade das vistas
  - Permite avaliar a qualidade das vistas. Este módulo é essencial para a escolha do melhor ponto de vista de cada célula, e do melhor ponto de vista de cada pintura exposta no museu virtual.
  - No que se refere ao processamento das imagens correspondentes às vistas, este módulo colabora com o módulo imagem.
- Visita guiada
  - Gera a visita guiada, que integra o caminho de alto nível e o caminho de baixo nível. O caminho de alto nível e o caminho de baixo nível são criado pelo módulo dos algoritmos de pesquisa.
  - O melhor ponto de vista de cada célula e o melhor ponto de vista de cada uma das pinturas são determinados em colaboração com o módulo qualidade das vistas.
  - O módulo visita guiada determina ainda a orientação da câmara ao longo dos vários pontos do caminho de baixo nível.

- Modelo de *voxels*
  - É responsável pela organização do modelo em *voxels* e pela atribuição do valor distância a cada um desses *voxels*.
  - Após se ter procedido à representação do modelo em *voxels*, este módulo organiza o modelo em células e portais.
- Museu
  - Responsável pela interface com o utilizador.
- Modelo
  - Permite a visualização do modelo do museu de pintura no ecrã. Além disso, lida com o posicionamento do utilizador no museu.
  - Este módulo tem um papel centralizador no protótipo, coordenando a interacção com os módulos museu, edifício, visita guiada, modelo de *voxels*, e qualidade das vistas.

Na Figura 3.1 são representados os módulos do protótipo, e as dependências existentes entre eles.

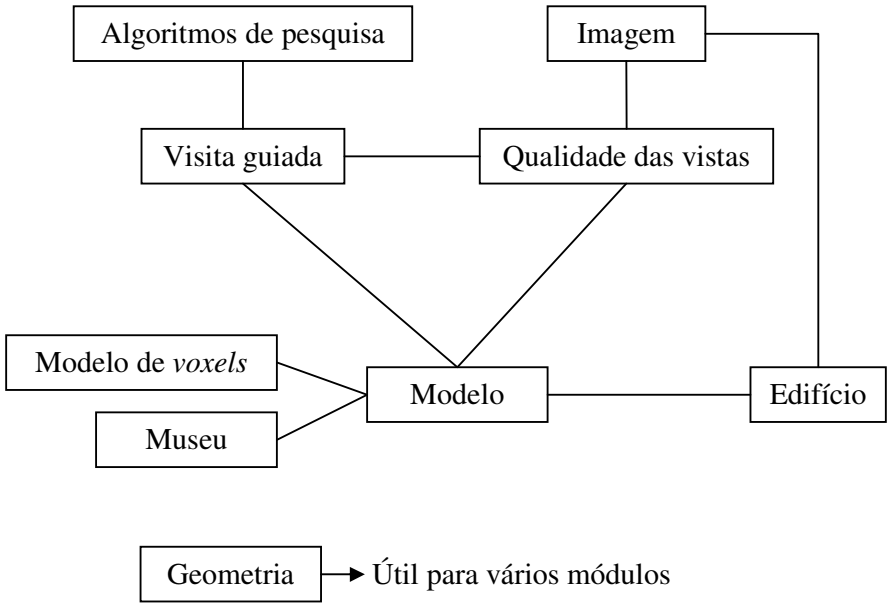


Figura 3.1: Módulos do programa

Seguidamente, mostramos os resultados do protótipo para o modelo de uma sala com algumas pinturas expostas no Museu Nacional de Arte Antiga. O modelo da sala é visível na Figura 3.2. As pinturas foram criadas na Oficina de Nuno Gonçalves, no século XV [Circ05,

IPM06]. No modelo podemos ver os Painéis de São Vicente de Fora, no topo. À esquerda encontram-se as pinturas São Teotónio e Santo Franciscano. À direita temos São Vicente Atado à Coluna [IPM06].

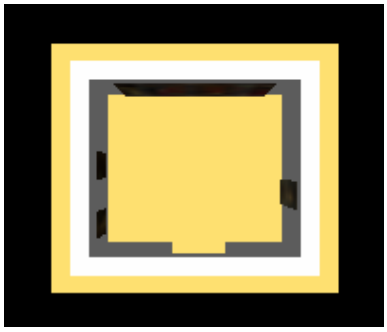


Figura 3.2: Sala com algumas pinturas do Museu Nacional de Arte Antiga

Na Figura 3.3 pode observar-se o corte horizontal da grelha de *voxels* resultante da organização em células e portais do modelo da Figura 3.2. Em cada *voxel* é indicado o número da célula a que esse *voxel* pertence. Caso o *voxel* intersecte uma pintura, é marcado pela letra 'P'. Os *voxels* correspondentes às paredes têm o número -1.

Repare-se que, para além de célula onde se encontram expostas as pinturas, o protótipo detectou ainda 6 outras células e que são as células 0, 1, 3, 4 e 5. Estas células adicionais devem-se ao chão do modelo, que se prolonga para além dos limites da célula onde estão expostas as pinturas.

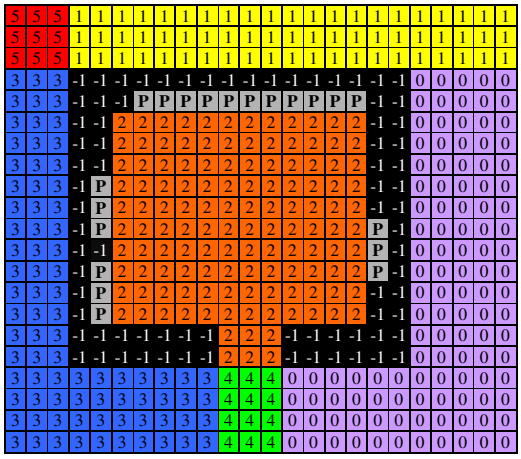


Figura 3.3: Corte horizontal da grelha de *voxels* demonstrativo da organização em células do modelo da Figura 3.2

Para evitar uma excessiva proliferação de células, optou-se por fundir as células que partilhem mais de 70% dos seus *voxels* de fronteira com outras células. Repare-se que é esse o caso das células 5 e 4. Sendo assim, após a fusão de células, a organização em células e portais é a

visível na Figura 3.4. A fusão das células 5 e 1 deu origem à célula 5. A fusão das células 4 e 0 deu origem à célula 4.

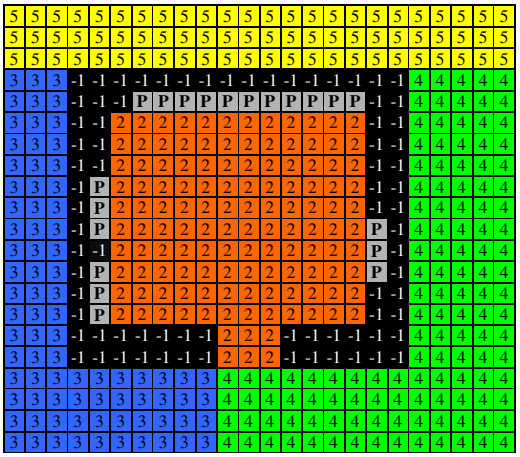


Figura 3.4: Corte horizontal da grelha de *voxels* demonstrativo da organização em células do modelo da Figura 3.2, depois da fusão de células

Após ter efectuado a organização em células e portais, o protótipo calcula a entropia dos pontos de vista apenas para os *voxels* que pertençam a células onde existem pinturas. No caso do modelo da Figura 3.2, somente a célula 2 contém pinturas.

Podia-se calcular a entropia para todos os *voxels* do modelo, mas isso iria aumentar o tempo de execução, não resultando daí nenhuma vantagem para o utilizador. Isto porque as células sem pinturas são, na prática, meros espaços vazios ou corredores. Este tipo de células não merece uma atenção especial, reduzindo-se a sua utilidade, para efeitos da visita guiada, a servir de passagem entre as células com pinturas.

Uma vez que a altura da câmara é constante, a entropia dos pontos de vista é calculada somente para a fatia horizontal do modelo de *voxels* que está a essa altura.

Na Figura 3.5 pode-se observar a representação do valor da entropia dos pontos de vista para cada um dos *voxels* da célula 2 do modelo da Figura 3.2. Quanto mais claro é o *voxel*, maior é a entropia e, portanto, maior a qualidade das vistas. Os *voxels* das células sem pinturas e os *voxels* intersectados por paredes são representados a negro uma vez que, para eles, a entropia não foi calculada.

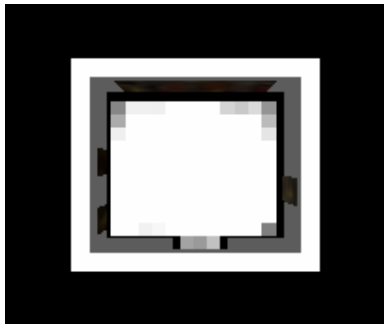


Figura 3.5: Entropia dos pontos de vista do modelo da Figura 3.2

Após se ter calculado a entropia dos pontos de vista para os *voxels* pertencentes a células com pinturas, o protótipo escolhe o melhor ponto de vista de cada célula com pinturas. Optámos por escolher apenas um único ponto de vista por célula, uma vez que estas são tendencialmente pequenas, sendo um único ponto vista suficiente para ter uma boa ideia geral acerca das mesmas.

Depois, escolhe-se o melhor ponto de vista para cada pintura. Para escolher o melhor ponto de vista para cada pintura, recorremos, também, à entropia dos pontos de vista. No cálculo da entropia, apenas se consideram as faces da pintura para a qual se pretende escolher o melhor ponto de vista. As faces dos restantes elementos do modelo, para efeitos do cálculo da entropia, contam apenas como fundo. Os pontos de vista candidatos para cada pintura, à semelhança do caso da escolha do melhor ponto de vista para cada célula, são os centros dos *voxels*, pertencentes a célula com pinturas, da fatia horizontal que está à altura da câmara.

É certo que a maximização da área projectada total poderia parecer um critério mais conveniente, pois maximizaria a área projectada das pinturas. No entanto, optámos pela entropia dos pontos de vista devido à importância dada ao fundo. Tal como já referimos anteriormente, o fundo ajuda o utilizador a compreender melhor a cena ou objecto em visualização. Ora, se a câmara se aproximar demasiado da pintura, de maneira a que mais nada seja visível à excepção da pintura, perde-se a noção do contexto e da dimensão da pintura.

Além disso, como refere o fotógrafo, pintor, desenhador, ilustrador, escultor e poeta Fernando Lemos, a generalidade das pessoas tem uma *"educação visual feita de livro"*, com a falsa percepção de que as pinturas são quase todas do mesmo tamanho. Isto porque nos livros, a dimensão das pinturas é reduzida ou aumentada, de forma a adaptar-se ao tamanho de uma página. Só através da visita aos museus é possível desfazer o equívoco dessa *"cultura visual de livro"*. Aí é possível contemplar, na sua verdadeira dimensão, os *"metros de parede"* que determinadas pinturas ocupam e o quanto outras, que julgávamos maiores, são afinal minúsculas [Lemo06]. Ora, justamente por estas razões, considerámos importante que a câmara permitisse ao utilizador, não apenas observar as pinturas, mas também aperceber-se da dimensão destas. Por isso, optámos pela entropia dos pontos de vista.

Seguidamente, o protótipo prossegue para a geração do caminho de alto nível. No caso do modelo da Figura 3.2, o caminho de alto nível é o mais simples possível, contendo apenas uma célula, a 2.

Criado o caminho de alto nível, o protótipo gera o caminho de baixo nível. Este caminho é suavizado com recurso a curvas.

Na Figura 3.6 mostra-se a visita guiada para o modelo da Figura 3.2. A azul é indicada a trajectória da câmara. O melhor ponto de vista da célula 2 é assinalado a vermelho. A verde, são indicados os melhores pontos de vista de cada pintura. As células 3, 4 e 5 não chegam sequer a ser visitadas porque não contêm pinturas, nem são úteis para o efeito de proporcionar passagem para outras células.

No melhor ponto de vista da célula, a câmara roda 360 graus de forma a mostrar ao utilizador o aspecto geral da área onde se encontra. A câmara pára também alguns momentos no melhor ponto de vista de cada pintura, de maneira a dar ao utilizador tempo para observar a obra em exposição.

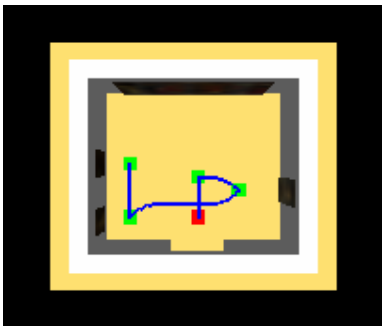


Figura 3.6: Visita guiada ao modelo da Figura 3.2

Na Figura 3.7 mostra-se uma imagem da visita guiada ao modelo da Figura 3.2, em que são visíveis as pinturas São Teotónio e Santo Franciscano [IPM06]. O protótipo permite ao utilizador escolher se o percurso da visita guiada e os melhores pontos de vista das células e das pinturas são mostrados ou ocultados.

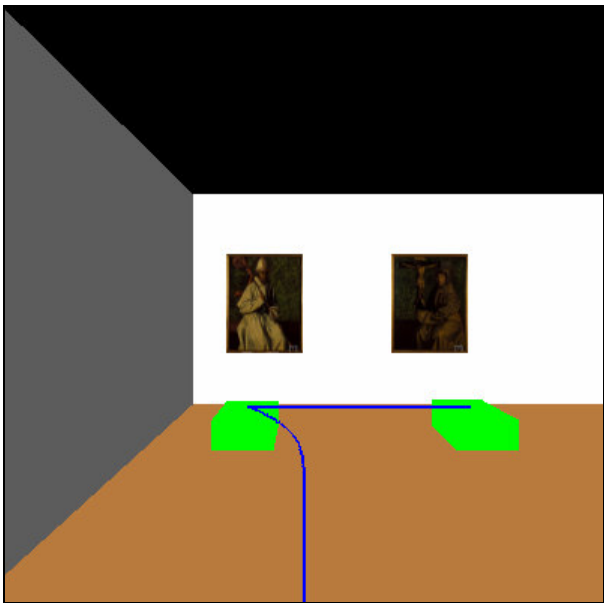


Figura 3.7: Imagem da visita guiada ao modelo da Figura 3.2

Na Figura 3.8 mostra-se a imagem correspondente ao melhor ponto de vista dos Painéis de São Vicente de Fora [IPM06] expostos no modelo da Figura 3.2.

Figura 3.8: Melhor vista dos Painéis de São Vicente de Fora expostos no modelo da Figura 3.2

No que se refere ao tempo de execução do protótipo, para o modelo da Figura 3.2, o Quadro 3.1 elucida-nos acerca deste aspecto. O computador utilizado para medir o tempo de execução do protótipo é um Pentium 4 de 3,4 GHz. As fases que duram mais tempo são a selecção do melhor ponto de vista para cada célula e a selecção do melhor ponto de vista para cada pintura. É natural que assim seja.

Na selecção do melhor ponto de vista para cada célula é necessário calcular a entropia para todos *voxels* da fatia horizontal que ficam à altura da câmara e que pertencem a células com pinturas. Para cada um desses pontos, é necessário processar 4 imagens, correspondentes às projecções que rodeiam o ponto de vista.

Podíamos processar as 6 projecções que rodeiam o ponto de vista, mas como não existem pinturas nem no chão nem no tecto isso é desnecessário e iria apenas tornar a duração de execução do protótipo mais morosa.

Ora, a leitura e processamento do *buffer* de cada uma dessas projecções com vista ao cálculo da visibilidade de cada face do modelo e, posteriormente, ao cálculo da entropia, é um processo moroso.

Na selecção do melhor ponto de vista para cada pintura é necessário, também, ler e processar o *buffer* das projecções e por isso a duração desta fase é elevada em relação às restantes.

Duração da execução das fases do protótipo para o modelo da Figura 3.2		
Fases de execução do protótipo	**Tempo (segundos)**	**Duração da fase (segundos)**
1. Organização do modelo em células e portais	1,750	1,750
2. Selecção do melhor ponto de vista para cada célula	11,156	9,406
3. Selecção do melhor ponto de vista para cada pintura	38,031	26,875
4. Criação do caminho de alto nível	38,046	0,015
5. Criação do caminho de baixo nível	40,640	2,594
Tempo total de execução (segundos)		*40,640*

Quadro 3.1: Duração da execução das fases do protótipo para o modelo da Figura 3.2

Para diminuir o tempo de execução do protótipo pode-se optar por aumentar o comprimento da aresta dos *voxels*, diminuindo dessa forma o número de *voxels*. Se existirem menos *voxels*, existem consequentemente menos pontos de vista para os quais a qualidade das vistas é avaliada. Só que um menor número de *voxels* aumenta a probabilidade de perder um excelente ponto de vista, que nunca chega a ser avaliado.

Outra forma de diminuir o tempo de execução consiste em reduzir a dimensão das imagens a serem lidas e processadas em cada ponto de vista. Quanto menor o número de pixels da imagem, mais rapidamente se calcula a entropia. Só que imagens muito pequenas podem fazer com que os pormenores do modelo deixem de ser visíveis. Sendo assim, esta não é também, uma estratégia isenta de desvantagens.

Há assim que tentar um equilíbrio entre uma escolha criteriosa dos melhores pontos de vista e o tempo de execução do protótipo. Neste, o comprimento da aresta dos *voxels* é de 0.50 e as imagens têm 400 pixels de largura por 400 pixels de altura.

Abordaremos agora os resultados do protótipo para um modelo de carácter mais complexo. Na Figura 3.9 podemos ver o modelo de um museu fictício.

Na Figura 3.10 pode observar-se o corte horizontal da grelha de *voxels* demonstrativo da organização em células do modelo da Figura 3.9. Na visível organização em células já se procedeu à fusão das que partilham mais de 70% dos seus *voxels* de fronteira com outras células. Por essa razão, as células 2, 5 e 7 desapareceram.

É de destacar, na Figura 3.10, uma grande área inacessível, do lado direito da célula 10 e do lado esquerdo da célula 11. Ora, esta área não é totalmente ocupada por paredes, tal com se pode confirmar na Figura 3.9. No entanto, a referida área foi considerada como sendo inacessível, porque não existem portais que lhe permitam o acesso.

O protótipo está, desta forma, preparado para eliminar as células que são inacessíveis para o utilizador, devido a não comunicarem com nenhuma outra.

A Figura 3.11 mostra a representação do valor da entropia dos pontos de vista para os *voxels* das células que contêm pinturas. Novamente, quanto mais claro é o *voxel*, maior é a entropia. Os *voxels* das células sem pinturas, e os *voxels* intersectados por paredes, são representados a negro pois, para eles, a entropia não foi calculada.

Figura 3.9: Museu fictício

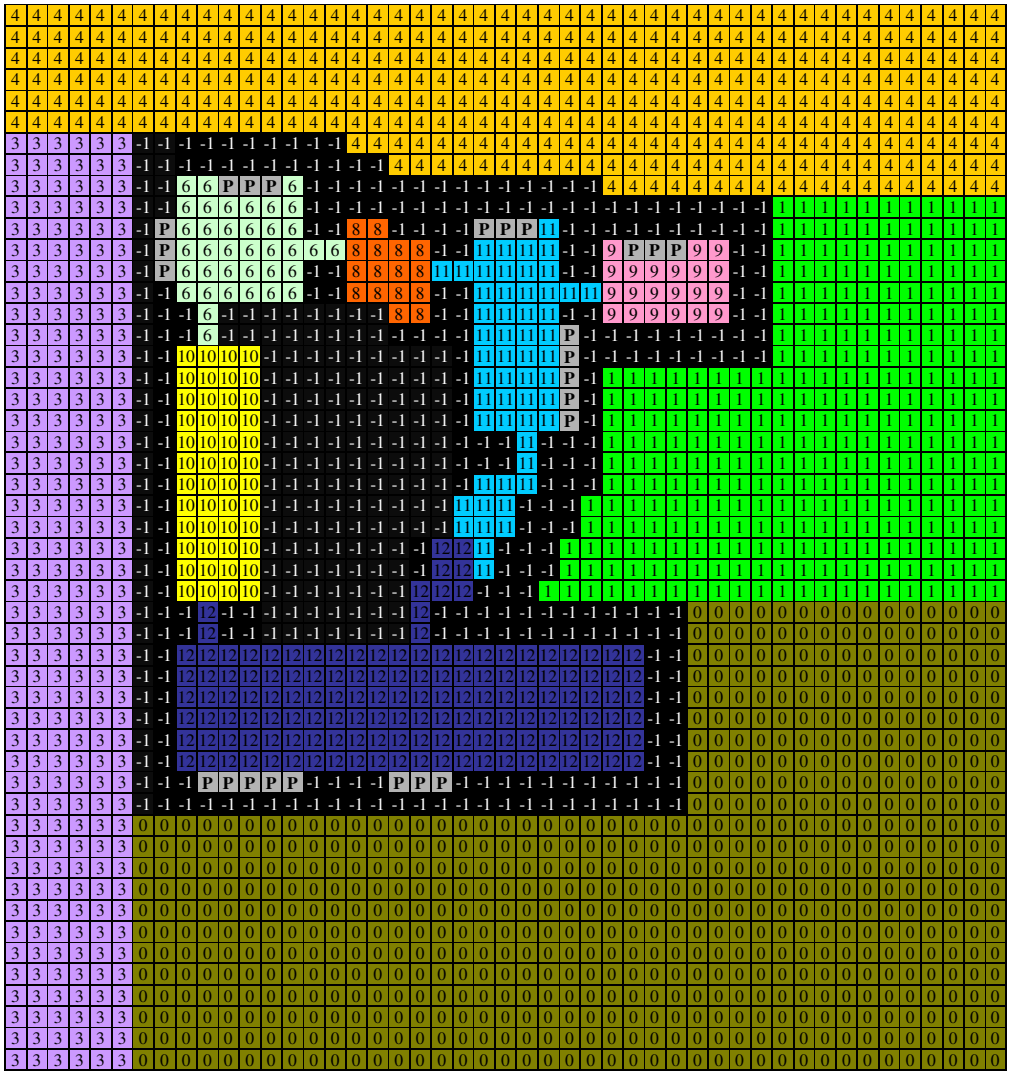


Figura 3.10: Corte horizontal da grelha de *voxels* demonstrativo da organização em células do modelo da Figura 3.9, depois da fusão de células

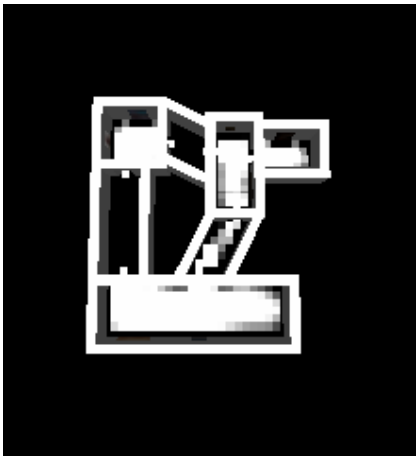


Figura 3.11: Entropia dos pontos de vista do modelo da Figura 3.9

Na Figura 3.12 pode observar-se a visita guiada ao modelo da Figura 3.9, com os melhores pontos de vista das células assinalados a azul e os melhores pontos de vistas das pinturas assinalados a verde. Repare-se que a célula 10 é incluída na visita apenas para permitir a passagem da célula 6 para a célula 12. Para a célula 10 não foi calculado o melhor ponto de vista porque esta célula não contém pinturas, não sendo, dessa forma, merecedora de uma atenção mais demorada.

É também de notar que, na visita às células com pinturas, a câmara começa sempre por dirigir-se ao melhor ponto de vista da célula de forma a, através dessa posição, poder dar ao utilizador uma boa ideia do espaço em que se encontra. Depois, a câmara dirige-se para a pintura que estiver mais próxima. Após essa pintura ter sido observada, dirige-se para a pintura que estiver mais próxima da que foi observada em primeiro lugar, e assim sucessivamente, até atingir a última pintura. Seguidamente a ter visualizado a última pintura, a câmara encaminha-se para a próxima célula.

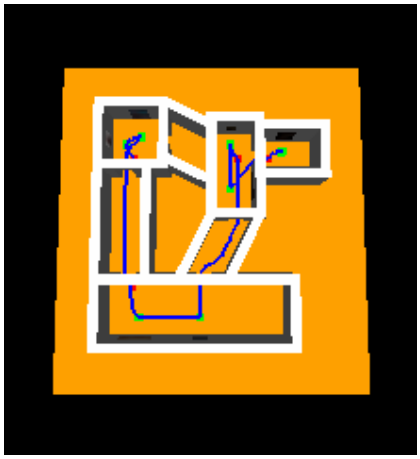


Figura 3.12: Visita guiada ao modelo da Figura 3.9

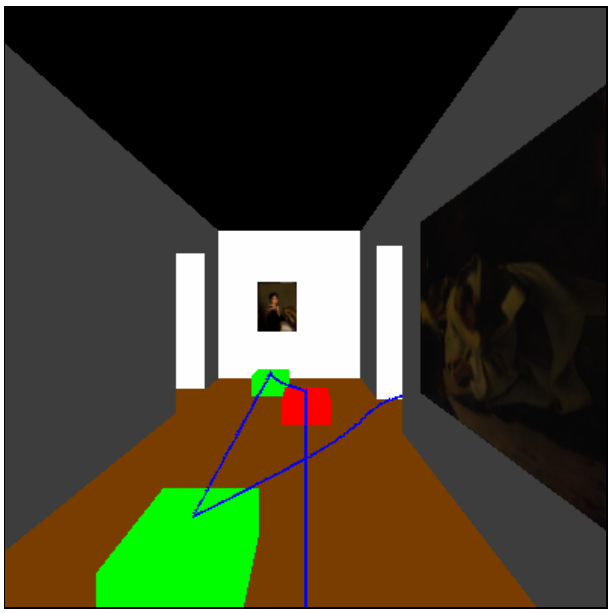


Figura 3.13: Imagem da visita guiada ao modelo da Figura 3.9

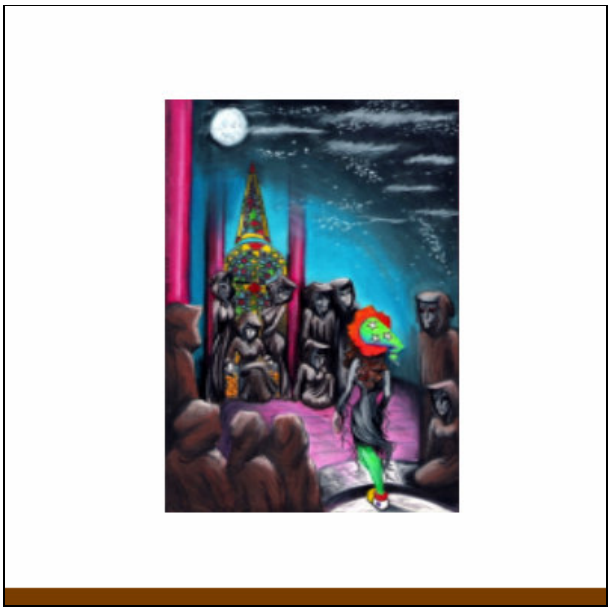


Figura 3.14: Melhor vista de uma das obras expostas na célula 6 do modelo da Figura 3.9

Na Figura 3.13 mostra-se uma imagem da visita guiada, ao modelo da Figura 3.9, na altura da visita à célula 11. A pintura à direita tem o título de São Bruno em Oração e é da autoria de Domingos António Sequeira [Circ05, IPM06]. Ao fundo encontra-se o Retrato de Mariana Benedita Vitória de Sequeira, pintura também de Domingos António Sequeira (Mariana Benedita Vitória de Sequeira é a filha do pintor).

Na Figura 3.14 encontra-se a melhor vista de uma das obras expostas na célula 6 do modelo da Figura 3.9.

No Quadro 3.2 indica-se a duração da execução das fases do protótipo para o modelo da Figura 3.9, num Pentium 4 de 3,4 GHz. Como este modelo contém mais células e mais pinturas do que o modelo da Figura 3.2, o tempo total de execução é mais longo.

Duração da execução das fases do protótipo para o modelo da Figura 3.9		
Fases de execução do protótipo	**Tempo (segundos)**	**Duração da fase (segundos)**
1. Organização do modelo em células e portais	2,765	2,765
2. Selecção do melhor ponto de vista para cada célula	23,937	21,172
3. Selecção do melhor ponto de vista para cada pintura	54,031	30,094
4. Criação do caminho de alto nível	54,046	0,015
5. Criação do caminho de baixo nível	67,359	13,313
Tempo total de execução (segundos)		*67,359*

Quadro 3.2: Duração da execução das fases do protótipo para o modelo da Figura 3.9

Há ainda que referir que o objectivo deste protótipo não é substituir uma visita real ao museu, mas sim proporcionar à pessoa uma previsão de como essa visita real irá decorrer. O protótipo pode também ser utilizado como uma forma alternativa de expor pinturas reais ou imagens criadas no computador.

4. Conclusão

Nesta dissertação, propôs-se estudar algoritmos que permitam efectuar uma visita guiada a um modelo volumétrico. Particularmente, assumiu-se que o modelo seria um espaço fechado, um museu, onde estão expostos quadros.

Uma vez que o modelo é um espaço fechado, foi estabelecido como objectivo compartimentar, de forma automática, esse espaço em várias divisões, às quais se deu o nome de células, que comunicam entre si através de portais. A divisão em células e portais exige a organização do modelo em *voxels*, para os quais é calculado o valor distância. A partir da grelha dos *voxels* e do valor distância é possível detectar as células e portais através de um algoritmo que é uma adaptação do algoritmo de segmentação por barragens. Portanto, pode concluir-se que o objectivo referente à organização do modelo em células e portais foi concretizado, tendo sido estudadas e implementadas estratégias que concretizam esta tarefa de forma automática, sem exigência de esforço ou interacção por parte do utilizador.

Após o modelo ser organizado em células, pode-se considerar esse modelo não como um todo, mas célula a célula. Seguidamente, foi necessário decidir quais as partes do modelo a visitar. Mais especificamente, foi necessário decidir quais as células a visitar, e, dentro dessas células, saber qual o caminho a percorrer pelo utilizador. Para tal, começou-se por proceder ao estudo de estratégias que permitem avaliar a qualidade de um ponto de vista. Um ponto de vista com qualidade é aquele que fornece ao utilizador vistas de qualidade onde o objecto ou os objectos a observar podem ser reconhecidos.

Entre os vários algoritmos que permitem avaliar a qualidade de uma vista, deu-se preferência à entropia dos pontos de vista. A entropia dos pontos de vista demonstrou-se eficaz, permitindo ao utilizador reconhecer com elevado sucesso o objecto perante o qual se encontra.

A qualidade baseada na curvatura das superfícies também mostrou bons resultados, apresentando ao utilizador vistas da qualidade. No caso do reconhecimento de um copo e no caso do reconhecimento de um candelabro, a qualidade baseada na curvatura das superfícies superou até a entropia dos pontos de vista, apresentando vistas de maior qualidade (Figura 2.24). No entanto, o algoritmo escolhido foi, apesar disso, a entropia dos pontos de vista, devido à maior atenção dada aos detalhes. Além disso, como o modelo no qual se efectua a visita guiada é um museu apenas com pinturas, a curvatura das superfícies assume aqui menor importância. Se no museu estivessem expostos outros tipos de obras de arte, como, por exemplo, estátuas, então, ter-se-ia dado preferência à qualidade baseada na curvatura das superfícies.

Escolhido o algoritmo de avaliação da qualidade das vistas, pôs-se o problema de saber como escolher os melhores pontos de vista das células. Estudaram-se várias formas de resolver este problema, acabando por se destacar os cálculos sucessivos da entropia. Esta estratégia, tal como o próprio nome indica, recorre a cálculos sucessivos da entropia dos pontos de vista, retornando os melhores pontos de vista para cada célula e um valor de relevância para cada célula. Quanto mais relevante é uma célula, mais informação esta mostra ao utilizador. Se uma célula tem elevado valor de relevância, então essa célula deve ser visitada. Se tem baixo valor de relevância, então é desnecessário visitá-la.

Uma vez que o modelo por onde decorre a visita guiada é um museu de quadros, optámos por outra estratégia, que consiste em visitar apenas as células que contém pinturas. Para essas células, através dos cálculos sucessivos da entropia, são escolhidos os melhores pontos de vista.

Portanto, a escolha dos melhores pontos de vista do modelo foi também outro objectivo concretizado, conseguindo-se que este processo fosse efectuado de forma automática.

No entanto, embora a entropia dos pontos de vista, em conjunção com os cálculos sucessivos da entropia, consigam encontrar as melhores vistas, não conseguem, infelizmente, distinguir um objecto do outro. Ou seja, a partir da informação geométrica, estes algoritmos não conseguem distinguir uma pintura de uma mancha na parede. Não percebem a diferença entre uma vulgar cadeira e uma estátua.

Portanto, através da aplicação destes algoritmos, a mancha na parede e a cadeira recebem a mesma informação que a pintura e a estátua, respectivamente. Alias, corre-se até o risco de a mancha e a cadeira receberem maior atenção, caso tenham uma maior concentração de faces.

É certo que alguém poderá, previamente, marcar as faces pertencentes às obras de arte. No entanto, isso é uma redução no total automatismo que à partida se pretendia.

Ensinar o computador a reconhecer uma obra de arte é uma tarefa difícil. Afinal, nem sequer os seres humanos nascem com esse conhecimento. Para poder apreciar uma pintura ou uma escultura, uma pessoa tem de ter um mínimo de formação artística. Caso contrário, num museu, essa pessoa poderá sentir-se como um analfabeto ao observar um texto.

Após se saber quais as células a visitar, e quais os melhores pontos de vista dentro dessas células, há então que encontrar um caminho que passe por todos esses pontos de vista. O estudo dos algoritmos de pesquisa levou à concretização deste objectivo.

Optou-se pelo algoritmo de pesquisa por aprofundamento progressivo para determinar a ordem pela qual as células são visitadas. Este algoritmo combina as vantagens da pesquisa em largura primeiro e da pesquisa em profundidade primeiro.

Para gerar o caminho que, dentro das células, passa pelos melhores pontos de vista, recorreu-se ao algoritmo A*, uma estratégia de pesquisa informada que, através da distância em linha recta até ao próximo ponto de vista, consegue mais rapidamente encontrar uma solução.

O caminho correspondente aos pontos devolvidos pelo A* é depois suavizado com recurso a uma curva. Concluiu-se que a curva B-Spline era a melhor opção. A curva Kochaneck-Bartels apresentou-se também como uma forte possibilidade. Só que a curva Kochaneck-Bartels passa por todos os pontos e, em certos casos, isso pode levar a situações indesejáveis, obrigando o utilizador a curvar constantemente para a esquerda e para a direita (Figura 2.56). A curva B-Spline reduz este tipo de problema (Figura 2.57).

Ainda assim, os desvios desnecessários não são totalmente eliminados (Figura 2.58). Por isso, procedeu-se à criação de uma nova estratégia que reduz os desvios desnecessários no caminho percorrido pela câmara. Esta estratégia detecta quando determinados pontos no caminho podem configurar um segmento de recta. A partir dos segmentos de recta detectados é possível gerar um novo caminho em que os desvios desnecessários da câmara para a esquerda e para a direita são eliminados. O algoritmo de melhoramento de caminhos zela para que o novo caminho evite atravessar obstáculos arquitectónicos, tais como paredes.

Foi também elaborado, no âmbito desta dissertação, um protótipo que conduz o utilizador numa visita guiada a um museu de pintura. Este protótipo faz mais do que, simplesmente, mostrar as obras, como se se estivesse perante uma mostra de slides. As pinturas são exibidas no contexto do próprio museu, o que proporciona ao utilizador maior informação acerca destas. Para começar, o próprio tamanho da obra torna-se mais evidente. Além disso, a simulação do ambiente do museu também é importante para a forma como a obra é apreciada.

Não se pretende, no entanto, que este protótipo funcione como um substituto à própria visita ao museu. É sim, uma preparação para a visita, que permitirá à pessoa orientar-se melhor quando estiver no local real.

O protótipo também pode ser utilizado na visita a um museu puramente virtual, funcionando como uma forma alternativa de exibir pinturas reais ou imagens digitais criadas no computador.

4.1. Trabalho futuro

Como trabalho futuro poderia proceder-se ao melhoramento do protótipo, concretizando os seguintes objectivos:

- Permitir a exibição de outros objectos, para além de pinturas, no museu virtual. Como exemplos de outros objectos podemos mencionar as esculturas, peças de mobiliário, peças de cerâmica, etc.
- Catalogar os objectos expostos no museu virtual. No caso específico das pinturas, podia-se organizá-las por autor, pela época em foram criadas ou pelo tipo de materiais utilizados na sua criação. Isto permitiria que o utilizador escolhesse qual o tipo de objectos que pretende observar na sua visita guiada.
- Incluir uma descrição detalhada de cada objecto exposto. Esta descrição seria apresentada ao utilizador no momento em que este está a observar o objecto, tornando, dessa forma, a visita mais esclarecedora e rica.
- Permitir a definição, para cada objecto, de zonas de pormenor que devem ser observadas de forma individualizada. Por exemplo, numa pintura, após se ter proporcionado ao utilizador uma visão geral da mesma, podia-se focar a câmara em pormenores específicos, para os quais seria proporcionada uma descrição esclarecedora.
- Criar um algoritmo que permita calcular o melhor ponto de vista de cada pintura mais rapidamente.

4.2. Algumas considerações sobre o processo de elaboração desta dissertação

Escolhi uma dissertação no âmbito da Computação Gráfica porque foi uma área que, desde cedo, me interessou e acerca da qual desejava saber mais.

Infelizmente, na minha licenciatura, embora tenha aprendido muitas outras competências que me têm sido indispensáveis na minha vida profissional, o gosto e curiosidade acerca da Computação Gráfica ficou, em grande medida, por concretizar. Isto porque o ensino da Computação Gráfica, em Informática e Gestão de Empresas, no ISCTE, limitou-se a uma cadeira no final do 5º ano. Em tão pouco tempo, apesar da dedicação e das qualidades pedagógicas do Professor, foi impossível aprofundar os conhecimentos.

Por isso, foi com grande entusiasmo que, no Mestrado em Engenharia Informática, vi surgir a oportunidade de me especializar nesta área que tanto me agrada.

O caminho percorrido foi compensador, mas difícil, porque, devido à pouca preparação em Computação Gráfica que trazia, a nível da licenciatura, tive, em vários aspectos de começar do zero. Ora, é do conhecimento comum o quanto os primeiros passos são difíceis, sendo, por isso, de inestimável valor a ajuda e orientação do Professor Doutor Próspero dos Santos.

Agora, sinto que o objectivo que me levou a fazer esta tese foi atingido. Obtive novos conhecimentos na área da Computação Gráfica e, hoje, sinto-me capaz e confiante para poder integrar uma equipa de trabalho dentro desta área. No entanto, o caminho não está de forma nenhuma terminado, e no futuro, espero poder continuar a evoluir na Computação Gráfica.

5. Bibliografia

[Aken01] Tomas Akenine-Möller. Fast 3D triangle-box overlap testing. Journal of Graphics Tools, 6(1):29-33, 2001.

[Andú04] C. Andújar, P. Vázquez e M. Fairén. Way-Finder: guided tours through complex walkthrough models. Em Eurographics, Volume 23, Number 3, 2004.

[Barr00] Pierre Barral, Guillaume Dorme, e Dimitri Plemenos. Scene understanding techniques using a virtual camera. Em A. de Sousa e J. C. Torres, editores, Proc. Eurographics'00, short presentations, 2000.

[Burg89] Peter Burger e Duncan Gillies. Interactive Computer Graphics – Functional, Procedural and Device-Level Methods. Addison-Wesley, 1989.

[Chas78] Sylvan H. Chasen. Geometric Principles and Procedures for Computer Graphics Applications. Prentice Hall, Estados Unidos da América, 1978.

[Circ05] Lourdes Cirlot; Museus do Mundo – Museu Nacional de Arte Antiga; Planeta de Agostini; Espanha; 2005.

[Core97] Galeria de objectos do CorelDREAM 3D 8.0 – Versão 8.232. Corel Corporation, 1997.

[Fole00] James Foley, Andries van Dam, Steven Feiner e John Hughes. Computer Graphics – Principles and Practice – Second Edition in C. Addison Wesley Publishing Company Inc., Abril de 2000.

[GIT06] Large Geometric Models Archive of the Georgia Institute of Technology. http://www.cc.gatech.edu/projects/large_models/index.html.

[Grev04] George J. Grevera. The "dead reckoning" signed distance transform. Em Elsevier, Computer Vision and Image Understanding 95, páginas 317 a 333 , 2004.

[Haum03] D. Haumont, O. Debeir e F. Silion. Volumetric cell-and-portal generation. Em Eurographics, Volume 22, Number 3, 2003.

[Hear97] Donald Hearn e M. Pauline Baker. Computer Graphics – C Version. Segunda Edição. Prentice Hall, Estados Unidos da América, 1997.

[Hill01] F. S. Hill, Jr. Computer Graphics Using Open GL. Segunda Edição. Prentice Hall, Estados Unidos da América, 2001.

[Huan01] Jian Huang, Yan Li, Roger Crawfis, Shao-Chiung Lu e Shuh-Yuan Liou. A Complete Distance Field Representation. Apresentado na IEEE Visualization 2001. October 21 - October 26, 2001. San Diego Paradise Point Resort, San Diego.

[Huan98] Jian Huang, Roni Yagel, Vassily Filipov e Yair Kurzion. An Accurate Method for Voxelizing Polygon Meshes. Proceedings of the IEEE symposium on Volume Visualization, páginas 119 a 126, 1998.

[IPM00] Desdobrável do Museu Nacional de Arte Antiga. Instituto Português de Museus, Lisboa, 2000.

[IPM06] MatrizNet – Colecções dos Museus IPM. Instituto Português de Museus. http://matriznet.ipmuseus.pt. 2006.

[Isto96] P. Isto. Path planning by multiheuristic search via subgoals. Em Proceedings of the 27th International Symposium on Industrial Robots, páginas 721 a 726, 1996.

[Isto97] P. Isto. A two-level search algorithm for motion planning. Em Proceedings IEEE International Conference on Robotics, páginas 2025 a 2031, 1997.

[Jone01a] M. W. Jones e R. A. Satherley. Shape Representation Using Space Filled Sub-Voxel Distance Fields. Em Proceedings of the International Conference on Shape Modeling & Applications, IEEE, página 316, USA, Whashington, 2001.

[Jone01b] Mark W. Jones e Richard Satherley. Using Distance Fields for Object Representation and Rendering. Em Proceedings of the 19th annual Conference of Eurographics (UK Chapter), London, páginas 37 a 44, 2001.

[Jone01c] Mark W. Jones e Richard Satherley. Vector-City Vector Distance Transform. Em Computer Vision and Image Understanding, Volume 82, Issue 3, páginas 238 a 254, Junho, 2001.

[Jone96] Mark W. Jones. The Production of Volume Data from Triangular Meshes Using Voxelisation. Em Eurographics, Volume 15, Number 5 páginas 311 a 318, 1996.

[Kall03] Marcelo Kallmann, Amaury Aubel, Tolga Abaci, e Daniel Thalmann. Planning collision-free reaching motions for interactive object manipulation and grasping. Computer Graphics Forum, 22(3), 2003.

[Kauf87a] A. Kaufman. An algorithm for 3D scan-conversion of polygons. Em Eurographics 87, páginas 197 a 208. North-Holland, Agosto 1987.

[Kauf87b] Arie Kaufman. Efficient Algorithms for 3D Scan-Conversion of Parametric Curves, Surfaces, and Volumes. Em Computer Graphics, Volume 21, Number 4, Julho 1987.

[Klei00] Konrad Klein e Vítor Sequeira. The View-Cube: An Efficient Method of View Planning for 3D Modelling from Range Data. Em Institute of Electrical and Electronics Engineers (IEEE), Fith IEEE Workshop on Applications of Computer Vision, Proceedings, 2000. Los Alamitos, Calif.: IEEE Computer Society, 2000.

[Lefe03] Sylvain Lefebvre e Samuel Hornus. Automatic Cell-and-portal Decomposition - Thème 3 - Interaction homme-machine, images, données, connaissances. Projets Evasion et Artis. Rapport de recherche n° 4898. Unité de recherche INRIA Rhône-Alpes, França. Julho de 2003

[Lemo06] Fernando Lemos. Entrevista ao fotógrafo Fernando Lemos por Carlos Vaz Marques no Programa Pessoal e… Transmissível. TSF. http://www.tsf.pt. 21 de Março de 2006.

[Lern03] Alon Lerner, Yiorgos Crysanthou e Daniel Cohen-Or. Breaking the Walls: Scene Partitioning and Portal Creation. Em Pacific Graphics, Canadá, 2003.

[Li00] T. Y. Li e H. K. Ting. An intelligent user interface with motion planning with 3d navigation. In Proc. IEEE VR, 2000.

[Mac03] David J. C. MacKay. Information Theory, Inference, and Learning Algorithms. Em Cambridge University Press, Reino Unido, 2003.

[Nied04] Christoph Niederberger, Dejan Radovic, Markus Gross. Generic path planning for real-time applications. Computer Graphics International, Proceedings: 299-306, 2004.

[Pear84] Judea Pearl. Heuristics – Intelligent Search Strategies for Computer Problem Solving. Addison-Wesley Publishing Company, 1984.

[Plem04] Dimitri Plemenos, Mateu Sbert e Miquel Feixas. On viewpoint complexity of 3D scenes. International Conference GraphiCon'2004, Moscovo, Rússia, Setembro de 2004.

[Rich91] Elaine Rich, Kevin Knight. Artificial Intelligence. McGraw-Hill, 1991.

[Russ03] Stuart Russell e Peter Norvig. Artificial Intelligence – A Modern Approach, Segunda Edição. Prentice Hall, Estados Unidos da América, Upper Saddle River, New Jersey, 2003.

[Sá03] Ana Sá e Bento Louro. Análise Matemática I-A – Teoria e Exercícios. http://ferrari.dmat.fct.unl.pt/services/AnaliseMatIA0506/. Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, Monte da Caparica, 2003.

[Salo03] Brian Salomon, Maxim Garber, Ming C. Lin, e Dinesh Manocha. Interactive navigation in complex environments using path planning. Em Proceedings of the 2003 symposium on Interactive 3D graphics, páginas 41 a 50, ACM Press, 2003.

[Sant06] Manuel Próspero dos Santos. Apontamentos das aulas de Computação Gráfica. http://ctp.di.fct.unl.pt/~ps/cg/. Universidade Nova de Lisboa, Monte de Caparica, 2006.

[Sber02] M. Sbert, M. Feixas, J. Rigau, F. Castro e P. P. Vázquez. Applications of Information Theory to Computer Graphics. Em Proceedings of 5th International Conference on Computer Graphics and Articial Intelligence (3IA'02), páginas 21a 36, Limoges, France, 2002.

[Sber05] Mateu Sbert, Dimitri Plemenos, Miquel Feixas, Francisco Gonzalez. Viewpoint Quality: Measures and Applications. Computational Aesthetics in Graphics, Visualization and Imaging, páginas 1 a 8, Girona, May 2005.

[Schn03] Philip J. Schneider e David H. Eberly. Goemetric Tools for Computer Graphics. Morgan Kaufman Publishers, Estados Unidos da América, São Francisco, 2003.

[Shan48] E.C. Shannon. A mathematical theory of communication. The Bell System Technical Journal, 27:379–423,623–656, July-October 1948.

[Shre04] Dave Shreiner, Mason Woo, Jackie Neider e Tom Davis. OpenGL Programming Guide – Fourth Edition – The Official Guide to Learning OpenGL, version 1.4. Addison-Wesley. 2004.

[Soko05] Dmitry Sokolov e Dimitri Plemenos. Viewpoint quality and scene understanding. The 6th International Symposium on Virtual Reality, Archaeology and Curtural Heritage – VAST 2005. Editores: M. Mudge, N. Ryan e R. Scopigno. 2005.

[Soko06] Dmitry Sokolov, Dimitri Plemenos e Karim Tamine. Viewpoint quality and global scene exploration strategies. International Conference on Computer Graphics Theory and Applications (GRAPP 2006), Portugal, Setúbal, 25 a 28 de Fevereiro de 2006.

[Stan05] The Stanford 3D Scanning Repository. http://www-graphics.stanford.edu/data/3Dscanrep/3Dscanrep.html.

[Tell91] Seth J. Teller e Carlo Séquin. Visibility preprocessing for interactive walkthroughs. In Proc. Computer Graphics (SIGGRAPH'91), volume 25, páginas 61 a 69. ACM Press, 1991.

[Vázq01a] Pere-Pau Vázquez, Miquel Feixas, Mateu Sbert e Wolfgang Heidrich. Viewpoint selection using viewpoint entropy. Em T. Ertl, B. Girod, G. Greiner, H. Niemann, e H.-P. Seidel, editores, Vision, Modeling, and Visualization 2001, 2001.

[Vázq01b] Pere Pau Vázquez, Miquel Feixas, Mateu Sbert e Wolfang Heidrich. Viewpoint Selection using Viewpoint Entropy. Em Proceedings of Vision, Modeling, and Visualization, Alemanha, 2001.

[Vázq02a] Pere-Pau Vazquez, Miquel Feixas, Mateu Sbert e Antoni Llobet. Viewpoint entropy: A new tool for obtaining good views of molecules. Em I. Navazo, P. Brunet, e D. Ebert, editors, Proceedings of Computer Graphics International 2002. ACM - SIGGRAPH, 2002.

[Vázq02b] Pere-Pau Vázquez e Mateu Sbert. Automatic Keyframe Selection for High-Quality Image-Based Walkthrough Animation Using Viewpoint Entropy. Em International Conference of Central Europe on Computer Graphics, Visualization and Computer Vision, Plzen, Fevereiro de 2002.

[Vázq03a] Pere Pau Vázquez. On the selection of good views and its application to computer graphics – Phd Dissertation, Espanha, Barcelona, 2 de Abril de 2003.

[Vázq03b] Pere-Pau Vázquez, Miguel Feixas, Mateu Sbert e Wolfang Heidrich. Automatic View Selection Using Viewpoint Entropy and its Application to Image-Based Modelling. Em Eurographics, Volume 22, Number 4, páginas 689-700, 2003.